QUESTION BANK

Name of the Department : **Computer Science and Engineering**

Subject Code & Name : **CS8391/Object Oriented Programming**

Year & Semester : **II & III**

UNIT I

Part -A

1. What are the advantages of object oriented programming over procedure oriented programming?

1)OOPs makes development and maintenance easier where as in Procedure-oriented programming language
it is not easy to manage if code grows as project size grows.
2)OOPs provides data hiding whereas in Procedure-oriented programming language a global data can be accessed
from anywhere.
3)OOPs provides ability to simulate real-world event much more effectively. We can provide the solution of real
word problem if we are using the Object-Oriented programming language.

2. Define abstraction?

Abstraction is the process used to hide certain details and only show the essential features of the object. It deals with the outside view of an object (interface)

3. Explain Encapsulation in java?

Encapsulation in java is a *process of wrapping code and data together into a single unit.*
We can create a fully encapsulated class in java by making all the data members of the Classprivate. Now we can use setter and getter methods to set and get the data in it.
The **JavaBean** class is the example of fully encapsulated class.

4. Define class?

A class is a user defined blueprint or prototype from which objects are created. It

represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

1. **Modifiers** : A class can be public or has default access
2. **Class name:** The name should begin with a initial letter (capitalized by convention).
3. **Superclass(if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends.
   A class can only extend (subclass) one parent.
4. **Interfaces(if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
5. **Body:** The class body surrounded by braces, { }.

5. Define object?

It is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, interact by invoking methods. An object consists of :

1. **State** : It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior** : It is represented by methods of an object. It also reflects the response of an object with other
   objects.
3. **Identity** : It gives a unique name to an object and enables one object to interact with other objects.

6. Explain the difference between class and object?

No.	Object	Class
1)	Object is an instance of a class.	Class is a blueprint or template from which objects are created.
2)	Object is a real world entity	Class is a group of similar objects.
3)	Object is a physical entity.	Class is a logical entity.
4)	Object is created through new keyword	Class is declared using class keyword e.g. class Student{}
5)	Object is created many times as per requirement.	Class is declared once.
6)	Object allocates memory when it is created.	Class doesn't allocated memory when it is created.
7)	There are many ways to create object in java such as new keyword, newInstance() method, clone() method, factory method and deserialization.	There is only one way to define class in java using class keyword.

7.Define Inheritance?

An important feature of object-oriented programs is inheritance - the ability to create classes that share the attributes and methods of existing classes, but with more specific features. Inheritance is mainly used for code reusability.

8.Explain Polymorphism in Java?

Polymorphism (from Greek, meaning "many forms") is a feature that allows one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation.

Polymorphism in Java has two types: Compile time polymorphism (static binding) and Runtime polymorphism (dynamic binding). Method overloading is an example of static polymorphism, while method overriding is an example of dynamic polymorphism.

9. Define Static polymorphism in java?

In Java, static polymorphism is achieved through method overloading. Method overloading means there are several

methods present in a class having the same name but different types/order/number of parameters.

At compile time, Java knows which method to invoke by checking the method signatures. So, this is called

compile time polymorphism or **static binding**

10. Explain Run time Polymorphism?

Run time Polymorphism also known as method overriding. In this Mechanism a call to an overridden function

is resolved at a run-time.

11. List the characteristics of java?

- Simple
- Secure
- Portable
- Object oriented
- Robust
- Multithreaded
- Architecture Neutral
- Interpreted

- High Performance
- Dynamic

12.How will you define class in java?

A class is a blueprint from which individual objects are created.

A class can contain any of the following variable types.

- **Local variables** − Variables defined inside methods, constructors or blocks are called local variables.

  The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.

- **Instance variables** − Instance variables are variables within a class but outside any method. These

  variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.

- **Class variables** − Class variables are variables declared within a class, outside any method, with the

  static keyword.

```
Class lamp
{
private boolean isOn;
publicvoid turnOn(){
 isOn =true;
}
Publicvoid turnOff()
{
        isOn =false;
}
}
```

13. Explain the purpose of constructor?

Constructors are used to initialize the object's state. Like methods, a constructor also contains **collection of statements** that are executed at time of Object Creation.

14.When is a Constructor called?

Each time an object is created using **new()** keyword at least one constructor (it could bedefault constructor) is invoked to assign initial values to the **data members** of the same Class. Constructor is invoked at the time of object or instance creation.

15. Explain the types of constructors in java?

There are two type of constructor in Java:

1. **No-argument constructor:** A constructor that has no parameter is known as default constructor. If we don't define a constructor in a class, then compiler creates **default constructor(with no arguments)** for the class.
2. **Parameterized Constructor:** A constructor that has parameters is known as parameterized constructor. If we want to initialize fields of the class with your own values, then use parameterized constructor.

16.How constructors are different from methods in Java?

- Constructor must have the same name as the class within which it defined while it is not necessary for the method in java.
- Constructor does not any return type while method(s) have the return type or **void** if does not return any value
- Constructor is called only once at the time of Object creation while method(s) can be called any numbers of time.

17.How will you create method in java ?

A Java method is a collection of statements that are grouped together to perform an Operation method definition consists of a method header and a method body. The same is shown in the following syntax −

Syntax

```
modifier  returnType  nameOfMethod (Parameter List) {
 // method body
}
```

- **modifier** − It defines the access type of the method and it is optional to use.
- **returnType** − Method may return a value.
- **nameOfMethod** − This is the method name. The method signature consists of the method name and the

  parameter list.
- **Parameter List** − The list of parameters, it is the type, order, and number of parameters of a method.

  These are optional, method may contain zero parameters.
- **Method body** − The method body defines what the method does with the statements.

18. List the access specifiers in Java?

Access Modifiers

1. Private
2. Protected
3. Default
4. Public

19.What is the use of Public Access Modifier

Fields, methods and constructors declared public (least restrictive) within a public

class are visible to any class in the Java program, whether these classes are in the

Same package or in another package.

20.What is the use of Private Access Modifier?

The private (most restrictive) fields or methods cannot be used for classes and

Interfaces. It also cannot be used for fields and methods within an interface. Fields,

Methods or constructors declared private are strictly controlled, which means they cannot be accesses by

anywhere outside the enclosing class. A standard design strategy is to make all fields private and provide public

getter methods for them.

21. What is the use of protected Access Modifier?

The protected fields or methods cannot be used for classes and Interfaces. It also cannot be used for fields and

methods within an interface. Fields, methods and constructors declared protected in a superclass can be accessed

only by subclasses in other packages. Classes in the same package can also access protected fields, methods and

constructors as well, even if they are not a subclass of the protected member's class.

22. Explain Default Access Modifier in java?

Java provides a defaultspecifier which is used when no access modifier is present. Any class, field, method or

constructor that has no declared access modifier is accessible only by classes in the same package.

The default modifier is not used for fields and methods within an interface.

23. Explain the purpose of java static variable?

- The static variable can be used to refer the common property of all objects (that is not unique for each

  object

- The static variable gets memory only once in class area at the time of class loading.

1) Advantage of static variable

It makes your program **memory efficient** (i.e. it saves memory).

24. Explain java static method?

- A static method belongs to the class rather than object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- Static method can access static data member and can change the value of it.

25. Explain the usage of Java packages.

This is a way to organize files when a project consists of multiple modules. It also helps resolve naming conflicts

when different packages have classes with the same names. Packages access level also allows you to protect data

from being used by the nonauthorized classes.

26.List the primitive data types in java?

TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS
boolean	true or false	false	1 bit	true, false
byte	twos complement integer	0	8 bits	(none)
char	unicode character	\u0000	16 bits	'a', '\u0041', '\101', '\\', '\'','\n',' β'
short	twos complement integer	0	16 bits	(none)
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d

27.Define variable and explain the declaration of variable in java ?

A variable provides us with named storage that our programs can manipulate.

Each variable in Java has a specific type, which determines the size and layout of the variable's memory;

the range of values that can be stored within that memory; and the set of operations that can be applied to the

variable.

You must declare all variables before they can be used. Following is the basic form of a variable declaration −

```
data type variable [ = value][, variable [ = value] ...] ;
```

Here *data type* is one of Java's datatypes and *variable* is the name of the variable. To declare more than one

variable of the specified type, you can use a comma-separated list.

Following are valid examples of variable declaration and initialization in Java

Example

```
int a, b, c;// Declares three ints, a, b, and c.
int a =10, b =10;// Example of initialization
```

28.Mention the use of control flow statements ?

Control flow statements, break up the flow of execution by employing decision making, looping, and

branching, enabling your program to *conditionally* execute particular blocks of code. The control flow statements

are `if-then`, `if-then-else`, `switch`, the looping statements (`for`, `while`, `do-while`), and the branching statements

(`break`, `continue`, `return`) supported by the Java programming language.

29. What is an Array in java ?

An array is a very common type of data structure wherein all elements must be of the same data type. Once defined,

the size of an array is fixed and cannot increase to accommodate more elements.The first element of an array starts

with index zero.

30. Explain the different ways of using arrays ?

1) Declaring Array

2) Constructing Array

3) Initialize Array

2) 1) Declaring Array

Syntax

<elementType>[] <arrayName>;

OR

<elementType><arrayName>[];

Example:

```
int intArray[];
 // Defines that intArray is an ARRAY variable which will store integer values
int []intArray;
```

3) 2) Constructing an Array

arrayname = new dataType[]

Example:

intArray = new int[10]; // Defines that intArray will store 10 integer values

Declaration and Construction combined

int intArray[] = new int[10];

4) 3) Initialize an Array

intArray[0]=1; // Assigns an integer value 1 to the first element 0 of the array

intArray[1]=2; // Assigns an integer value 2 to the second element 1 of the array

Part B

1 i. Explain OOPS and its features.

ii.Summarize about the usage of constructor with anexample using Java.

2 i. What is class? How do you define a class in Java?

ii. Examine the use of inheritance and class hierarchy.

3 i. Define polymorphism with example program.

ii. Describe variables and operators in Java.

4. Define and explain the control flow statements in Java with suitable examples

5 What is meant by constructor? Discuss the types of constructor with example.

6. i.Analyse and Develop a simple Java program to sort the given numbers in increasing order.

ii.Write a Java program to reverse the given number.

7 i. Classify the characteristics of Java.

ii.Illustrate the working principles of Java Virtual Machine.

iii.Show with an example the structure of Java Program

8 i.Summarize about access specifier in Java.

ii.Describe the term static fields and methods and explain itstypes with example.

9 i) Define Arrays. What is array sorting and explain with an example.

ii)Tabulate and explain documentation comments in Java.

10 Illustrate what is meant by package? How its types are created and implemented in Java.

UNIT II

1.Explain the advantages of inheritance?

Inheritance is one of the cornerstones of object-oriented programming because it allows the creation of hierarchical

classifications. Using inheritance, you can create a general class that defines traits common to a set of related items

2. Define SuperClass?

In the terminology of Java, a class that is inherited is called a superclass

3. Define Subclass?

The class that does the inheriting is called a subclass. Therefore, a subclass is a specialized version of a superclass.

It inherits all of the members defined by the superclass and adds its own, unique elements.

4. Explain the use of Protected Access Modifier

Variables, methods, and constructors, which are declared protected in a superclass, can be accessed only by the

subclasses in other package or any class within the package of the protected members' class.

The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared

protected, however methods and fields in a interface cannot be declared protected.

Protected access gives the subclass a chance to use the helper method or variable, while preventing a nonrelated

class from trying to use it.

5.What is the use of super keyword in java?

The **super** keyword in java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred

by super reference variable.

Usage of java super Keyword

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.

6. How does a subclass class call superclass constructors?

A subclass can call a constructor defined by its superclass by use of the following form of super:

super(arg-list);

Here, arg-list specifies any arguments needed by the constructor in the superclass.

super( ) must always be the first statement executed inside a subclass' constructor.

7.Define Object class in Java?

Object class is present in java. Lang package. Every class in Java is directly or indirectly derived from the Object

class. If a Class does not extend any other class then it is direct child class of Object and if extends other class then

it is an indirectly derived. Therefore the Object class methods are available to all Java classes. Hence Object class

acts as a root of inheritance hierarchy in any Java Program.

8. Explain the methods in object class?

Object clone( ) Creates a new object that is the same as the object being cloned.

boolean equals(Object object) Determines whether one object is equal to another.

void finalize( ) Called before an unused object is recycled.

Class getClass( ) Obtains the class of an object at run time.

int hashCode( ) Returns the hash code associated with the invoking object.

void notify( ) Resumes execution of a thread waiting on the invoking object.

void notifyAll( ) Resumes execution of all threads waiting on the invoking object.

String toString( ) Returns a string that describes the object.

void wait( ) void wait(long milliseconds)

void wait(long milliseconds, int nanoseconds) Waits on another thread of execution

9. Explain the purpose of abstract class?

Certain methods be overridden by subclasses by specifying the abstract type modifier. These methods are

sometimes referred to as subclasser responsibility because they have no implementation specified in the

superclass. Thus, a subclass must override them.

To declare an abstract method, use this general form:

abstract type name(parameter-list);

No method body is present. Any class that contains one or more abstract methods must also be declared abstract.

To declare a class abstract, you simply use the abstract keyword in front of the class keyword at the beginning of

the class declaration.

10. List the properties of abstract class ?

There can be no objects of an abstract class. That is, an abstract class cannot be directly instantiated with the new

operator. Such objects would be useless, because an abstract class is not fully defined. Also, you cannot declare

abstract constructors, or abstract static methods. Any subclass of an abstract class must either implement all of the

abstract methods in the superclass, or be declared abstract itself.

11.Define abstract method?

A method without body (no implementation) is known as abstract method. A method must always be declared in

an abstract class,

12. What are the rules of abstract method?

1. Abstract methods don't have body, they just have method signature.
2. If a class has an abstract method it should be declared abstract, the vice versa is not true, which means an abstract class doesn't need to have an abstract method.
3. If a regular class extends an abstract class, then the class must have to implement all theabstract methods of

abstract parent class or it has to be declared abstract as well.

13. Mention the use of final method?

A final method cannot be overridden. Which means even though a sub class can call the final method of parent

class without any issues but it cannot override it.

```
class XYZ
{
final void demo(){
     System.out.println("XYZ Class Method");
}
}

class ABC extends XYZ{
  public static void main(String args[]){
     ABC obj= new ABC();
     obj.demo();
  }
}
```

14.Mention the purpose of final class ?

We cannot extend a final class. Consider the below example:

```
final class XYZ{
}

class ABC extends XYZ{
  void demo(){
     System.out.println("My Method");
  }
  public static void main(String args[]){
     ABC obj= new ABC();
     obj.demo();
  }
```

}
The type ABC cannot subclass the final class XYZ

15. Define the properties of interfaces ?

Using the keyword interface, you can fully abstract a class' interface from its implementation. That is, using

interface, you can specify what a class must do, but not how it does it. Interfaces are syntactically similar to classes

but they lack instance variables, and their methods are declared without any body

Once it is defined, any number of classes can implement an interface. Also, one class can implement any number

of interfaces. To implement an interface, a class must create the complete set of methods defined by the interface.

However, each class is free to determine the details of its own implementation. By providing the interface keyword

Java allows you to fully utilize the "one interface, multiple methods" aspect of polymorphism. Interfaces are

designed to support dynamic method resolution at run time

16. How will you define an interface?

An interface is defined much like a class. This is a simplified general form of an interface:

Access interface name

{

return-type method-name1(parameter-list);

return-type method-name2(parameter-list);

type final-varname1 = value;

type final-varname2 = value;

}

When no access modifier is included, then default access results, and the interface is only available to other

members of the package in which it is declared. When it is declared as public, the interface can be used by any

other code

Variables can be declared inside of interface declarations. They are implicitly final and static, meaning they cannot

be changed by the implementing class. They must also be initialized. All methods and variables are implicitly

public.

17. How will you implement an interface ?

Once an interface has been defined, one or more classes can implement that interface.

To implement an interface, include the implements clause in a class definition, and then create the methods defined

by the interface.

The general form of a class that includes the implements clause looks like this:

class classname [extends superclass] [implements interface [,interface...]]

{ // class-body }

If a class implements more than one interface, the interfaces are separated with a comma

18.Explain the difference between class and interface?

1. A class can be instantiated by creating its objects. An interface is never instantiated as the methods declared

5) inside an interface are abstract and does not perform any action, so there is no use of instantiating any

6)interface.

2. A class is declared using a keyword class. In the same way, an interface is created using a keyword interface.
3. The members of a class can have the access specifier like public, private, protected. But the members of an

interface are always public as they have to be accessed by the classes implementing them.

4. The methods inside a class are defined to perform an action on the fields declared in the class. As interface

   lacks in the declaration of fields, the methods inside an interface are purely abstract.

5. A class can implement any number of interfaces but can extend only one super class. An interface can extend

   any number of interfaces but cannot implement any interface.

6. A class has constructors defined inside it to get the variable initialized. But, an interface does not have any

   constructors as there are no fields to be initialized. The fields of an interface are initialized at the time of

   their declaration only.

19. How will you extend the interfaces ?

An interface can extend another interface in the same way that a class can extend another class. The extends

keyword is used to extend an interface, and the child interface inherits the methods of the parent interface

20. Why do we use object cloning in java?

The object cloning is a way to create exact copy of an object. The clone() method of Object class is used to clone

an object.

The java.lang.Cloneable interface must be implemented by the class whose object clone we want to create.

If we don't implement Cloneable interface, clone() method generates CloneNotSupportedException.

The clone() method is defined in the Object class. Syntax of the clone() method is as follows:

protected Object clone() **throws** CloneNotSupportedException

21. Define inner class?

Inner class or nested class is a class which is declared inside the class or interface.

We use inner classes to logically group classes and interfaces in one place so that it can bemore readable and

maintainable.

Additionally, it can access all the members of outer class including private data members and methods.

Syntax of Inner class

```
class Java_Outer_class{
//code
class Java_Inner_class{
//code
}
}
```

22. Explain arraylist in java?

ArrayList is a part of collection framework and is present in java.util package. It provides us dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed.

- ArrayList inherits AbstractList class and implements List interface.
- ArrayList is initialized by a size, however the size can increase if collection grows or shrunk if objects are removed from the collection.
- Java ArrayList allows us to randomly access the list.
- ArrayList can not be used for primitive types, like int, char, etc. We need a wrapper class for such cases (see this for details).
- ArrayList in Java can be seen as similar to vector in C++

23. Define String in java ?

String is a sequence of characters. But in java, string is an object that represents a sequence of characters. The java.lang.String class is used to create string object.

24. How to create String object?

There are two ways to create String object:

1. By string literal
2. By new keyword

25. List the different types of inheritance in java ?

The different types of inheritance which is supported by Java.

- Single Inheritance
- Multiple Inheritance **(Through** Interface**)**
- Multilevel Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance **(Through Interface)**

Part-B

1 i. **Describe** in detail **about** inheritance.
ii. Write a program for inheriting a class.
2 i**. Illustrate** what is super and subclass in Java .
ii. With an example, illustrate how the objects from sub class are inherited by the super class.
3 **Examine** how to control top level and member level access for the members of the class.
4 **Design** with an example how passing objects as parameters to methods and returning objects From methods in Java
5 **Describe** in brief about object class and its methods in Java with suitable example.
6 i i. **Explain** with an example what is meant by object cloning?
ii. **Summarize** in detail about inner class with its usefulness.
7 i**. Explain** briefly on final keyword.
ii**. Explain** the concept of abstract class with an example.
8 i. **Describe** what is meant by interface.
ii.How is interface declared and implemented in Java. Give example.
9 **i. Differentiate c**lasses with interface with suitable examples.
ii. **Express** a Java program for extending interfaces.
10 **i. Define** inner classes. How to access object state using inner classes? Give an example.

UNIT –III
Part - A

1. Define Exception ?

An exception is an abnormal condition that arises in a code sequence at run time. In other words, an exception is a
runtime error. In computer languages that do not support exception handling, errors must be checked and handled
manually—typically through the use of error codes.

2. List the keywords used to handle exception ?

Java exception handling is managed via five keywords: try, catch, throw, throws, and finally.

3. Mention the use of try block ?

Program statements that you want to monitor for exceptions are contained within a try block.
If anexception occurs within the try block, it is thrown. Your code can catch this exception
(using catch) and handle it in some rational manner.

4. What is the use of throw and throws clause ?

System-generated exceptions are automatically thrown by the Java runtime system. To manually throw an
exception, use the keyword throw. Any exception that is thrown out of a method must be specified as such by a
throws clause

5. Specify the use of finally block?

Any code that absolutely must be executed after a try block completes is put in a finally block

6. Write the general form of an exception-handling block ?

```
try { // block of code to monitor for errors }
catch (ExceptionType1 exOb)
{ // exception handler for ExceptionType1 }
catch (ExceptionType2 exOb) { // exception handler for ExceptionType2 }
// ... finally { // block of code to be executed after try block ends }
```

Here, ExceptionType is the type of exception that has occurred

7. Explain exception class hierarchy?

All exception types are subclasses of the built-in class Throwable. Thus, Throwable is at the top of the exception class hierarchy. Immediately below Throwable are two subclasses that partition exceptions into two distinct branches. One branch is headed by Exception. This class is used for exceptional conditions that user programs should catch.

This is also the class that you will subclass to create your own custom exception types. There is an important
subclass of Exception, called RuntimeException. Exceptions of this type are automatically defined for the programs
that you write and include things such as division by zero and invalid array indexing.

The other branch is topped by Error, which defines exceptions that are not expected to be caught under normal
circumstances by your program. Exceptions of type Error are used by the Java run-time system to indicate errors
having to do with the run-time environment, itself. Stack overflow is an example of such an error.

8. Specify the use of multiple catch Clauses ?

In some cases, more than one exception could be raised by a single piece of code. To handle this type of situation,
you can specify two or more catch clauses, each catching a different type of exception. When an exception is
thrown, each catch statement is inspected in order, and the first one whose type matches that of the exception is
executed. After one catch statement executes, the others are bypassed, and execution continues after the try / catch
block.

9. Give examples for Unchecked exception ?

ArithmeticException - Arithmetic error, such as divide-by-zero.
ArrayIndexOutOfBoundsException- Array index is out-of-bounds.
ArrayStoreException- Assignment to an array element of an incompatible type.
ClassCastException -Invalid cast.
EnumConstantNotPresentException -An attempt is made to use an undefined enumeration value.
IllegalArgumentException- Illegal argument used to invoke a method.
IllegalMonitorStateException- Illegal monitor operation, such as waiting on an unlocked thread.
IllegalStateException- Environment or application is in incorrect state.
IllegalThreadStateException- Requested operation not compatible with current thread state. I
ndexOutOfBoundsException- Some type of index is out-of-bounds.
NegativeArraySizeException -Array created with a negative size.

NullPointerException -Invalid use of a null reference.
NumberFormatException- Invalid conversion of a string to a numeric format.
SecurityException- Attempt to violate security.
StringIndexOutOfBounds- Attempt to index outside the bounds of a string.
TypeNotPresentException- Type not found.
UnsupportedOperationException- An unsupported operation was encountered.

10. Give examples of checked exceptions in java ?

ClassNotFoundException- Class not found.
CloneNotSupportedException- Attempt to clone an object that does not implement the Cloneable interface.
IllegalAccessException- Access to a class is denied.
InstantiationException -Attempt to create an object of an abstract class or interface.
InterruptedException- One thread has been interrupted by another thread.
NoSuchFieldException -A requested field does not exist.
NoSuchMethodException- A requested method does not exist.
ReflectiveOperationException- Superclass of reflection-related exceptions.

11. How to create own exceptions?

Define a subclass of Exception (a subclass of Throwable). subclasses don't need to actually implement anything
it is their existence in the type system that allows you to use them as exceptions. The Exception class does not
define any methods of its own. It does, of course, inherit those methods provided by Throwable. Thus, all
exceptions, including those that we create, have the methods defined by Throwable available to them.

12. Define StacktraceElement class ?

The StackTraceElement class describes a single stack frame, which is an individual element of a stack trace when
an exception occurs. Each stack frame represents an execution point, which includes such things as the name of the
class, the name of the method, the name of the file, and the source-code line number.

An array of StackTraceElements is returned by the getStackTrace( ) method of the Throwable class.
StackTraceElement has one constructor:

StackTraceElement(String className, String methName, string fileName, int line)

Here, the name of the class is specified by className, the name of the method is specified in methName, the name

of the file is specified by fileName, and the line number is passed in line. If there is no valid line number,
use a negative value for line. Furthermore, a value of –2 for line indicates that this frame refers to a native method.

13.Describe the methods of StacktraceElement class ?

boolean equals(Object ob)- Returns true if the invoking StackTraceElement is the same as the one passed in ob.
Otherwise, it returns false.
String getClassName( )- Returns the name of the class in which the execution point described by the invoking
StackTraceElement occurred.
String getFileName( ) -Returns the name of the file in which the source code of the execution point described by
the invoking StackTraceElement is stored.
int getLineNumber( )- Returns the source-code line number at which the execution point described by the invoking
StackTraceElement occurred. In some situations, the line number will not be available, in which case a negative
value is returned.
String getMethodName( ) -Returns the name of the method in which the execution point described by the invoking
StackTraceElement occurred.
int hashCode( ) -Returns the hash code for the invoking StackTraceElement.
boolean isNativeMethod( ) -Returns true if the execution point described by the invoking StackTraceElement
occurred in a native method. Otherwise, it returns false.
String toString( ) - Returns the String equivalent of the invoking sequence.

14.Define Stream ?

Java programs perform I/O through streams. A stream is an abstraction that either produces or consumes
information. A stream is linked to a physical device by the Java I/O system

15. Define bytestream ?

Byte streams provide a convenient means for handling input and output of bytes. Byte streams are used, for example, when reading or writing binary data.
Byte streams are defined by using two class hierarchies. At the top are two abstract classes: InputStream and OutputStream.

16. Define CharacterStream ?

Character streams provide a convenient means for handling input and output of characters. They use Unicode and, therefore, can be internationalized. Also, in some cases, character streams are more efficient than byte streams.

Character streams are defined by using two class hierarchies. At the top are two abstract classes: Reader and Writer. These abstract classes handle Unicode character streams.

17. Mention the use of bufferedReader object ?

In Java, console input is accomplished by reading from System.in. To obtain a character based stream that is attached to the console, wrap System.in in a BufferedReader object. BufferedReader supports a buffered input stream.
A commonly used constructor is shown here:

BufferedReader(Reader inputReader)

Here, inputReader is the stream that is linked to the instance of BufferedReader that is Beingcreated. Reader is an abstract class.

18. How to read character from input stream ?

To read a character from a BufferedReader, use read( ). The version of read( ) that we will be using is

int read( ) throws IOException Each time that read( ) is called, it reads a character from the input stream and returns it as an integer value.

19. How to read strings from keyboard ?

To read a string from the keyboard, use the version of readLine( ) that is a member of the BufferedReader class. Its general form is shown here:

String readLine( ) throws IOException

20. How to perform Console output ?

Console output is most easily accomplished with print( ) and println( ).These methods are defined by the class PrintStream

Because PrintStream is an output stream derived from OutputStream, it also implements the low-level method write( ). Thus, write( ) can be used to write to the console.

The simplest form of write( ) defined by PrintStream is shown here:

void write(int byteval)

This method writes the byte specified by byteval. Although byteval is declared as an integer

21.Explain printwriter class ?

The recommended method of writing to the console when using Java is through a PrintWriter stream.

PrintWriter is one of the character-based classes.

PrintWriter defines several constructors.

PrintWriter(OutputStream outputStream, boolean flushOnNewline)

Here, outputStream is an object of type OutputStream, and flushOnNewline controls whether Java flushes the output stream every time a println( ) method is called. If flushOnNewline is true, flushing automatically takes place. If false, flushing is not automatic.

22.Define FileInputstream ?

FileInputStream is useful to read data from a file in the form of sequence of bytes

FileInputStream is meant for reading streams of raw bytes such as image data. For reading streams of characters, consider using FileReader.

Constructor and Description

- **FileInputStream(File file) :**Creates an input file stream to read from the specified File object.
- **FileInputStream(FileDescriptor fdobj) :**Creates an input file stream to read from the specified file descriptor.
- **FileInputStream(String name) :**Creates an input file stream to read from a file with the specified name.

23. Define FileOutputStream ?

Java FileOutputStream is an output stream used for writing data to a file.

If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as

well as character-oriented data through FileOutputStream class. But, for character-oriented data, it is preferred to

use FileWriter than FileOutputStream.

24. Write a program that uses write( ) to output the character "A" followed by a newline to the screen

```
public static void main(String args[])
{
```

```
int b;
b = 'A';
System.out.write(b);
System.out.write('\n'); } }
```

25.**Write a program to handle division by zero ?**

```
class Exc2 {
public static void main(String args[])
{
int d, a;
try {
d = 0;
a = 42 / d;
catch (ArithmeticException e) {
System.out.println("Division by zero."); }
System.out.println("After catch statement."); } }
```

Part -B

1 Discuss in detail about exception handling constructs and write a program to illustrate Divide by zero exception.
2 Describe the following concepts with example
i. Try-catch-throw paradigm.
ii.Exception specification.
3 Describe about the syntax of catch and re-throw an exception with an example
4 i)Point out the importance of exception hierarchy.
ii) Explain on stack trace elements give example
5 Tabulate any five classes to support exception handling in Java with an example for each.
6 i. Summarize what is finally class. How to catch exception? Write an example.
ii. Give short notes on Java built in exceptions
7 Analyse the following in detail with example program
i.Checked Exception

ii.Unchecked exception

8 i.Classify the errors and exception in Java.

ii.Illustrate about multiple catching exceptions with example.

9 Summarize the following with example program

i.Arithmetic exception

ii.Arrayoutofbound exception

ii.Stringindexoutofbound exception

10 i. Develop a program to read and count the characters from files.

ii.Develop a Java program to transfer the content of one file to another file.

UNIT IV

Part A

1. **Define Multithreading ?**

A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is

called a thread, and each thread defines a separate path of execution. Thus, multithreading is a specialized

form of multitasking.

2. **What is Multitasking ?**

Multitasking is the ability to perform more than one activity concurrently on a computer.

We can further break multitasking into process based and thread based .

3. **Explain the types of multitasking?**

Process-based Multitasking

Process-based multitasking allows processes (or programs) to run concurrently on a computer

Thread-based Multitasking

Thread-based multitasking allows different parts of the same process (program) to run Concurrently.

4. **Differentiate multithreading and multitasking ?**

1)In multitasking, several programs are executed concurrently while in multi-threading multiple threads execute either same or different part of program multiple times at the

same time.

2)Multi-threading is more granular than multi-tasking. In multi-tasking, CPU switches between multiple programs to complete their execution in real time, while in multi-threading CPU switches between multiple threads of the same program.
switching between multiple processes has more context switching cost than switching between multiple threads of the same program.

3) Process are heavyweight as compared to threads, they require their own address space, which means multi-tasking is heavy compared to multithreading. Inter-process communication is expensive and limited and context switching from one process to another is expensive and limited.

5. **Describe the different states of thread ?**

Threads exist in several states. A thread can be running. It can be ready to run as soon as it gets CPU time.

A running thread can be suspended, which temporarily halts its activity. A suspended thread can then be

resumed, allowing it to pick up where it left off. A thread can be blocked when waiting for a resource. At any

time, a thread can be terminated, which halts its execution immediately. Once terminated, a thread cannot be

resumed.

6. **Explain methods in thread class ?**

The Thread class defines several methods that help manage threads.

getName -Obtain a thread's name.

getPriority -Obtain a thread's priority.

isAlive -Determine if a thread is still running.

join -Wait for a thread to terminate.

run -Entry point for the thread.

sleep -Suspend a thread for a period of time.

start -Start a thread by calling its run method.

7. **Explain the different ways of creating thread?**

Thread can be created by instantiating an object of type Thread.

Java defines two ways in which this can be accomplished

• Implementing the runnable interface.

•By extending the thread class

8. How will you create thread by using runnable interface?

The easiest way to create a thread is to create a class that implements the Runnable interface.

Runnable abstracts a unit of executable code. You can construct a thread on any object that implements

Runnable.

To implement Runnable, a class need only implement a single method called run( ), which is declared like this

public void run( ) Inside run( ), you will define the code that constitutes the new thread.

9. How will you create thread by using thread class?

The second way to create a thread is to create a new class that extends Thread, and then to create an instance of that class. The extending class must override the run( ) method, which is the entry point for the new thread. It must also call start( ) to begin execution of the new thread.

10. Mention the use of thread priorities?

Thread priorities are used by the thread scheduler to decide when each thread should be allowed to run.

In theory, higher-priority threads get more CPU time than lower-priority threads.

11. How do we specify priority for thread ?

To set a thread's priority, use the setPriority( ) method, which is a member of Thread.

This is its general form: final void setPriority(int level)

Here, level specifies the new priority setting for the calling thread. The value of level must be within the range

MIN_PRIORITY and MAX_PRIORITY. Currently, these values are 1 and 10, respectively. To return a thread

to default priority, specify NORM_PRIORITY, which is currently 5. These priorities are defined as static final

variables within Thread.

12. Explain the life cycle of thread?

A thread goes through various stages in its life cycle. For example, a thread is born, started, runs, and then dies.

The following diagram shows the complete life cycle of a thread.

Following are the stages of the life cycle −

- **New** − A new thread begins its life cycle in the new state. It remains in this state until the program starts

  the thread. It is also referred to as a **born thread**.

- **Runnable** − After a newly born thread is started, the thread becomes runnable. A thread in this state is

  considered to be executing its task.

- **Waiting** − Sometimes, a thread transitions to the waiting state while the thread waits for another thread to

  perform a task. A thread transitions back to the runnable state only when another thread signals the

  waiting thread to continue executing.

- **Timed Waiting** − A runnable thread can enter the timed waiting state for a specified interval of time.

  A thread in this state transitions back to the runnable state when that time interval expires or when the

  event it is waiting for occurs.

- **Terminated (Dead)** − A runnable thread enters the terminated state when it completes its task or otherwise

  terminates.

13.Mention the need for synchronization ?

When two or more threads need access to a shared resource, they need some way to ensure that the resource will be used by only one thread at a time. The process by which this is achieved is called synchronization.

13. What is monitor ?

A monitor is an object that is used as a mutually exclusive lock. Only one thread can own a monitor at a given time. When a thread acquires a lock, it is said to have entered the monitor. All other threads attempting to enter the locked monitor will be suspended until the first thread exits the monitor. These other threads are said to be waiting for the monitor. A thread that owns a monitor can reenter the same monitor if it so desires.

14. How interprocess communication is implemented in java?

Java includes an elegant interprocess communication mechanism via the wait( ), notify( ), and notifyAll( ) methods. These methods are implemented as final methods in Object, so all classes have them. All three methods can be called only from within a synchronized context .

- wait( ) tells the calling thread to give up the monitor and go to sleep until some other thread enters the same monitor and calls notify( )
- notify( ) wakes up a thread that called wait( ) on the same object
- notifyAll( ) wakes up all the threads that called wait( ) on the same object. One of the threads will be granted access.

15. List the properties of daemon thread in java?

Daemon thread is a low priority thread that runs in background to perform tasks such as garbage collection.

Properties:

- They cannot prevent the JVM from exiting when all the user threads finish their execution.
- JVM terminates itself when all user threads finish their execution
- If JVM finds running daemon thread, it terminates the thread and after that shutdown itself. JVM does not care whether Daemon thread is running or not.
- It is an utmost low priority thread.

16. Differentiate daemon thread and user thread?

1. **Priority:** When the only remaining threads in a process are daemon threads, the interpreter exits. This
   makes sense because when only daemon threads remain, there is no other thread for which a daemon
   thread can provide a service.
2. **Usage:** Daemon thread is to provide services to user thread for background supporting task

17. Explain the concept of Threadgroup?

Thread groups provide a mechanism for collecting multiple threads into a single object and manipulating
those threads all at once, rather than individually. For example, you can start or suspend all the threads
within a group with a single method call. Java thread groups are implemented by the ThreadGroup class in the java.langpackage.

18. Mention the advantages of using generic programming?

Generics enable *types* (classes and interfaces) to be parameters when defining classes,
Interfaces and methods.

Type parameters provide a way for you to re-use the same code with different inputs

Code that uses generics has many benefits over non-generic code:

- Stronger type checks at compile time.

  A Java compiler applies strong type checking to generic code and issues errors if the code violates type

  safety. Fixing compile-time errors is easier than fixing runtime errors, which can be difficult to find.

- Elimination of casts.

19. Explain the general form of generic class?

The General Form of a Generic Class T

class class-name { { // …

class-name var-name = new class-name(cons-arg-list);

20. Explain the use of Generic Methods?

A single generic method declaration can be called with arguments of

Different types. Based on the types of the arguments passed to the generic method, thecompiler handles

each method call appropriately.

21. What Are Generics?

The term generics means parameterized types. Parameterized types are important because they enable you

to create classes, interfaces, and methods in which the type of data upon which they operate is specified as a

parameter.

Using generics, it is possible to create a single class, for example, that automatically works with different

types of data. A class, interface, or method that operates on a parameterized type is called generic, as in

generic class or generic method.

22. Specify the use of bounded types?

Java provides bounded types.When specifying a type parameter, you can create an upper bound that declares the superclass from which all type arguments must be derived. This is accomplished through the

use of an extends clause when specifying the type parameter, as shown here: This specifies that T can

only be replaced by superclass, or subclasses of superclass. Thus, superclass defines an inclusive, upper limit.

23. Give an example for generic constructors ?

```
private double val; GenCons(T arg)
 {
val = arg.doubleValue();
}
void showval()
{ System.out.println("val: " + val); }
}
```

24. Give an example to define generic interfaces ?

```
interface MinMax<T extends Comparable<T>>
{
```

```
T min(); T max();
}
```

25. What is the use of isalive() and join() ?

The isAlive( ) method returns true if the thread upon which it is called is still running. It returns false otherwise

While isAlive( ) is occasionally useful, the method that you will more commonly use to wait for a thread to finish is called join( ),

```
final void join( ) throws InterruptedException
```

This method waits until the thread on which it is called terminates. Its name comes from the concept of the calling thread waiting until the specified thread joins it. Additional forms of join( ) allow you to specify a maximum amount of time that you want to wait for the specified thread to terminate.

Part –B

1 **Describe i**n detail about multithread programming with example.

2 i.**Differentiate** multithreading and multitasking.

ii.**Describe** the properties of thread in detail.

3 **Summarize** the two types of thread implementation supported by Java .Give example.

4 i.**Illustrate** the concept of synchronization in thread.

ii.**Write** a Java code for reader writers problem.

5 i. **Describe** how to implement runnable interface for creating and starting threads.

ii.**Define** threads. Describe in detail about thread life cycle.

6 i.**Explain** what is inter-thread communication? List out the methods used for it.

ii.**Explain** inter-thread communication using producer consumer problem.

7 **Summarize** the following

i.Thread priorities

ii.Deamon thread

8 **Explain** the following

i) States of a thread with a neat diagram

ii) Explain how threads are created in Java

9i.**Formulate** the motivations of generic programming.

ii.**Develop** a program to show generic class and methods with example.

10 i.**Describe** in detail about bounded types with suitable example.

ii.**List** the inheritance rules for generic types with example.

UNIT V

Part A

1.Mention the advantages of using swing ?

- Swing has a rich and convenient set of user interface elements.
- Swing has few dependencies on the underlying platform; it is therefore less prone to Platform specific bugs.
- Swing gives a consistent user experience across platforms
- Swing programs can give the program a specific "look-and-feel.

2. How to Create a Frame?

A top-level window (that is, a window that is not contained inside another window) is called a frame in Java. The AWT library has a class, called Frame, for this top level. The Swing version of this class is called JFrame and extends the Frame class. The JFrame is one of the few Swing components that is not painted on a canvas.

3. Explain the methods for Positioning a Frame ?

- The setLocation and setBounds methods for setting the position of the frame
- The setIconImage method, which tells the windowing system which icon to display in the title bar,task switcher window, and so on

• The setTitle method for changing the text in the title bar
• The setResizable method, which takes a boolean to determine if a frame will be resizeable by the user.

4. How to set the Frame Properties ?

Many methods of component classes come in getter/setter pairs, such as the following methods of the Frame class:

public String getTitle()
public void setTitle(String title)
Such a getter/setter pair is called a property. A property has a name and a type.

5. Explain the types of pane in JFrame ?

There are four panes layered in a JFrame. The root pane, layered pane, glass pane(they are Required to organize the menu bar) and contentpane to implement the look-and-feel.

6. Give an example to construct point?

```
Point2D p = new Point2D.Double(10, 20);
```

7. Specify the syntax to draw the ellipse ?

Ellipse2D ellipse = new Ellipse2D.Double(centerX - width / 2, centerY - height / 2, width, height);

8. How will you draw the line ?

Line2D line = new Line2D.Double(startX, startY, endX, endY);

9. How to set color using graphics2D class ?

The setPaint method of the Graphics2D class lets you select a color that is used for all subsequent drawing operations on the graphics context.
For example:

```
g2.setPaint(Color.RED);
g2.drawString("Warning!", 100, 100);
```

10.What is the use of fill () method in graphics2D class ?

You can fill the interiors of closed shapes (such as rectangles or ellipses) with a color using fill()

```
Rectangle2D rect = . . .;
g2.setPaint(Color.RED);
 g2.fill(rect); // fills rect with red color
```

11.How to change background color and foreground color ?

Color getBackground()
void setBackground(Color c)
gets or sets the background color.

Color getForeground()
void setForeground(Color c)
gets or sets the foreground color.

12. Write a code that displays the string "Hello, World!" in the standard sans serif font on your system, using 14-point bold type

```
Font sansbold14 = new Font("SansSerif", Font.BOLD, 14);
g2.setFont(sansbold14);
String message = "Hello, World!";
g2.drawString(message, 75, 100);
```

13.Explain the syntax to create a new font and to draw image ?

Font(String name, int style, int size) creates a new font object.
boolean drawImage(Image img, int x, int y, ImageObserver observer)

14.Explain how event handling in the AWT works ?

- A listener object is an instance of a class that implements a special interface called a listener interface
- An event source is an object that can register listener objects and send them event objects.
- The event source sends out event objects to all registered listeners when that event Occurs
- The listener objects will then use the information in the event object to determine their reaction to the event.

15. Describe event object?

When something happens in the application, an event object is created. For example, when we click on the button or select an item from a list. There are several types of events, including ActionEvent, TextEvent, FocusEvent, and ComponentEvent.

An event object holds information about an event that has occurred.

16. Explain the different ways of implementing event handling?

There are several ways how we can implement event handling in Java Swing:

- Anonymous inner class

- Inner class
- Derived class

17. List the types of adapter class ?

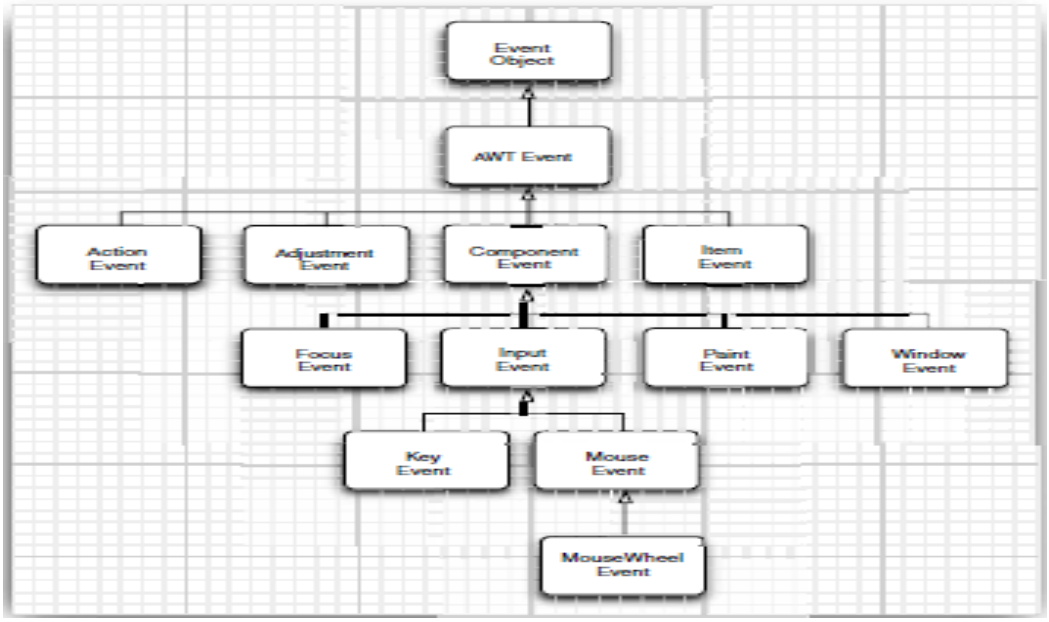
- ContainerAdapter class.
- KeyAdapter class.
- FocusAdapter class.
- WindowAdapter class.
- MouseAdapter class.
- ComponentAdapter class.
- MouseMotionAdapter class.

18. Explain mouse event handling ?

In order to handle mouse events, Mouse Listener has to be defined. Mouse listener is the handler which will handle mouse events. There are two types of mouse listener

1. **MouseListener**
   This listener is used to track click,press, release, entered and exited events.
2. **MouseMotionListener**
   This listener is used to track mouse movements, drag movements etc.

19. Describe the AWT event hierarchy ?



19. What is the purpose of MVC?

•The model, stores the content

• The view, displays the content

• The controller, handles user input

The advantages of the model-view-controller pattern is that a model can have multiple views, each showing a different part or aspect of the full content

20Specify the use of Border Layout ?

You can choose to place the component in the center, north, south, east, or west of the content pane

21. Mention the purpose of Grid Layout ?

The grid layout arranges all components in rows and columns like a spreadsheet.

All components are given the same size.

22 Give an example to create Textfield ?

```
JPanel panel = new JPanel();
JTextField textField = new JTextField("Default input", 20);
panel.add(textField);
```

23. Explain the methods to create TextArea in Swing ?

• JTextArea()

• JTextArea(int rows, int cols)

• JTextArea(String text, int rows, int cols)

constructs a new text area.

24. Differentiate modal and modeless dialog boxes ?

A modal dialog box won't let users interact with the remaining windows of the application until he or she deals

with it. Use a modal dialog box when you need information from the user before you can proceed with execution.

For example, when the user wants to read a file, a modal file dialog box is the one to pop up.

A modeless dialog box lets the user enter information in both the dialog box and the remainder of the application.

One example of a modeless dialog is a toolbar. The toolbar can stay in place as long as needed, and the user can

interact with both the application window and the toolbar as needed.

25. List the types of layouts in Swing?

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout
- GroupLayout
- SpringLayout

26. Differentiate between AWT and Swing ?

No.	Java AWT	Java Swing
1)	AWT components are platform-dependent.	Java swing components are platform-independent.
2)	AWT components are heavyweight.	Swing components are lightweight.
3)	AWT doesn't support pluggable look and feel.	Swing supports pluggable look and feel.
4)	AWT provides less components than Swing.	Swing provides more powerful components such as tables, lists, scrollpanes,

		colorchooser, tabbedpane etc.
5)	AWT doesn't follows MVC(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing follows MVC.

Part B

1 i.Describe in detail about working with 2D shapes in Java.

ii. Identify a Java program to illustrate Mouse Events.

2 i..Describe in detail about swing Components.

ii.Describe the types of layout management.

3 Summarize in detail about graphics programming.

4 i. Discuss how an application can respond to events in Java? Write the steps and the example.

ii.Discuss the adapter class using example.

5 What is meant by event handling? Analyse and write a simple calculator using mouse events that restrict only addition,subtraction, multiplication and division.

6 Tabulate the controller design pattern and components of swing briefly.

7 i. Illustrate what is layout management? State the various types of layout supported by Java? Which layout is default one?

ii. Examine the basic of event handling.

8 Evaluate with an example program and discuss in detail about Mouse listener and Mouse Motion Listener.

9 i. Formulate the methods available in graphics for COLOR.

ii. Design the methods available to draw shapes

10 i. Explain on AWT Event Hierarchy

ii. Explain about Semantic and Low-Level Events