<body> Don't forget
</hot>

10 Marks

Ans.

Compile time Polymorphism	Run-time Polymorphism
i) also called Static Polymorphism	i) also called Dynamic Polymorphism
ii) also called early binding	ii) also called late binding
iii) faster program execution	iii) Slower program execution
iv) Less flexible	iv) More flexible
v) Eg. Overloaded Method	v) Eg. Overridden Method

i) Method Overloading :-

i) More than one method shares some method name with a different signature in the class.
ii) Return type can or cannot be same
iii) Change in the parameter is MUST.

Date __/__/__

Method Overridding:-

i) Derived class provides the specific implementation of the method that is already provided by the base class or parent class.

ii) Return type must be same or Covariant

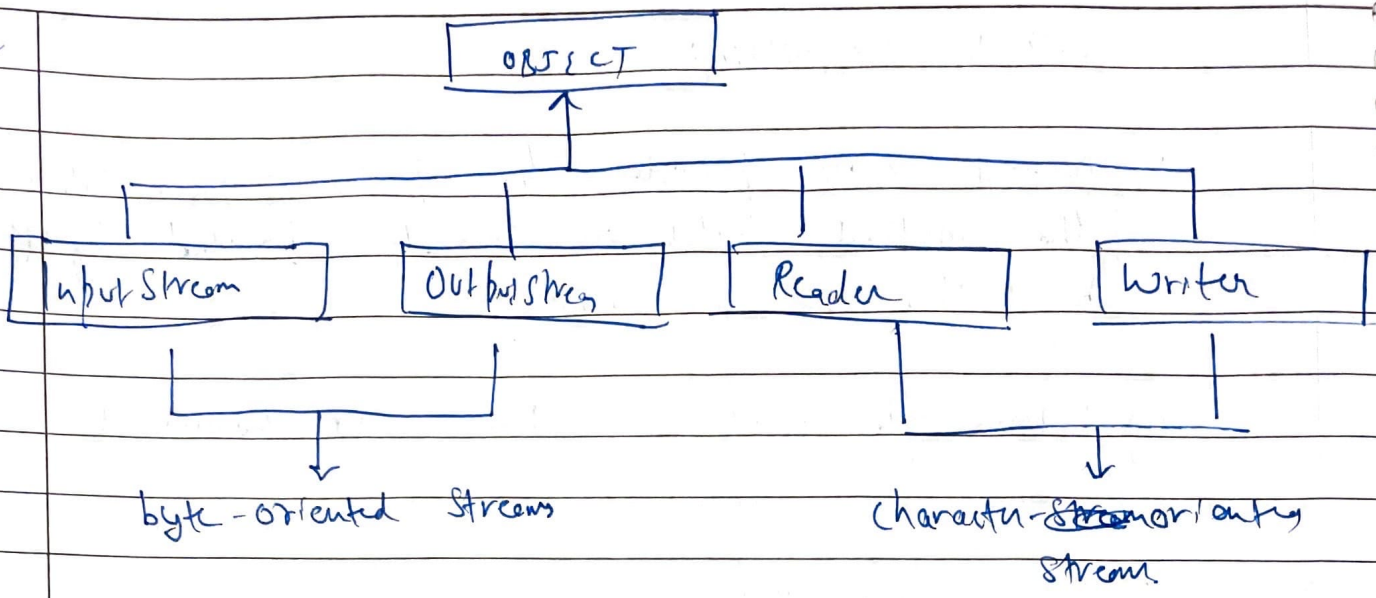
Program:-

```
import java.util.Arrays;
Public class Processor {

    Public void Process(int i, int j) {
        System.out.println("Processing two integer : "+i+", "+j);
    }
    Public void Process(int[] ints) {
        System.out.println("Adding integer Array " + Array.toString(ints));
    }
}
class MathProcessor extends Processor {
    @Override
    Public void process(int i, int j) {
        System.out.println("Sum of integer is " + (i+j));
    }
}.
```

Date __/__/__

Ans:



1) Byte-oriented Stream

- Intended for general-purpose input & output
- Data may be primitive data types or raw bytes

2) Character-oriented Stream

- Intended for character data
- Data is transformed from/to 16-bit java chars inside program to the UTF format used externally.

Program

```
import java.io.*

Public class FileInputStream {

    Public static void main (String [] args) throws
        FileNotFoundException, IOException {
```

Date __/__/__

```
String filename = "/home/jonbodna/tmp/bigfile.txt";

try (BufferedReader br = new BufferedReader(new InputStreamReader(
    new FileInputStream(filename), StandardCharsets.UTF_8))) {

String line;

while ((line = br.readLine()) != null) {

    System.out.println(line);
}
}
System.out.println();

}
}
```

Ans:- XML (Extensible Markup language) is a markup language similar to HTML, but without predefined tags to use. Instead, you define your own tags that can be stored, searched and shared. Most importantly since the fundamental format of XML is standardized, if you share or transmit XML across system or platform, either locally or over the internet, the recipient can still parse the data due to the standardized XML syntax.

Date __/__/__

Benefits of XML are:

1) XML Separate data from html:

In XML data can be stored in separate XML. This way we can focus on HTML/CSS for display & layout

2) XML Simplify data sharing :-

XML data is stored in plain text format This provides a software and hardware independent way of storing data.

3) XML Increases data availability :-

XML your data can be availble to all kind of 'Reading Machins (handheld Comptrs, voice machins, new feeds etc.) and mak it availble for blind people, or people with othr disability.

4) XML Can be used to Crat new Internet language

→ Alot of new langugs Can be creatn

1) XHTML.
2) RSS language for news feed
3) SMIL for multimedia

Date __/__/__

Program:-

```
<?XML version = "1.0" encoding = "UTF-8" ?>
<text>
<para> Hellow world </para>
</text>
```

Ans. - There are types of Access Modifier

i) Default - No keyword required
ii) Private
iii) Public
iv) Protected.

1) Defaut Program

i)
```
Packag P1;

class Geek
{
void display ()
{
  S.O.P ("Hellow world");
} }
```

ii)
```
Packag P2;
Import P1.*;

class GeekNew
{
  Public static void main (String args [])
{  Geek obj = new Geek ();
     obj.disp();
} }
```

Date __/__/__

ii) Private.

i)
```
Package P1;
Class A
{
Private void display ()
   {
      System.out.println ("Geek for Geek");
   }
}
class B
{
Public static void main (String args [])
      {
         A obj = new A();
         obj.display ();
      }
}
```

iii) Protected.

```
Package P1;
Public class A
{
Protected void display ()
   {
      System.out.println ("Geek for Geek");
   }
}
```

Date __/__/__

ii)
```
Package p2;
import p1.*;

Class B extends A
{
Public Static Void main (String args [])
    B obj = new B ();
     obj . display ();
}
}
```

iv) Public

i)
```
Package P1;
Public class A
{
 Public ~~Void~~ Void display ()
{
   System.out.Println ("GeeksforGeeks");
}
}
```

ii)
```
Package P2;
import P1.*;
class B {
Public Static void main (String args[])
{
    A obj = new A ();
      obj.display ();
  }
}
```

Date _/_/_

Ans. There are two ways of creating thread.

i) By extending Thread class
ii) By implementing Runnable interface

Thread class:- It provide constructors and methods to create and perform operations on a thread. Thread class extend object class & implements runnable interface.

Runnable interface :- The runnable should be implemented by any class whose instances are intended to be executed by a thread.

i)
```
class Multi extends Thread {
Public void run()
{System.out.println("Thread is running...");
}
Public static void main (String args[]){
Multi t1 = new Multi();
t1.start();
}
}
```

ii)
```
class Multi3 implements Runnable.{
Public void run() {
System.out.println("Thread is running...");
}
```

Date __/__/__

```
Public Static Void main (String args [])[
MultiT3 m1 = new MultiT 3();
Thread t1 = new Thread (m1)
t1.start ();
}
}
```

→ Thread Synchronization

```
Class Table {
Synchonized void PrintTable (int n) {
    for (int i = 1 = 1; i <= 10; i++)
        System.out.println (n + "*" + i + "=" + i*n);
}
}

Class MyThread_1 extends Thread {
    Table table = new Table ();
    int numbr;

    MyThread_1 (Table table, int numbr) {
        this.table = table;
        this.numbr = numbr;
    }
    Public void run () {
        table.PrintTable(numbr);
    }
}
Class My_Thread_2 extend Thread {
    Table tabl = new Table ();
    int numb;
```

Date __/__/__

```
MyThread_2 (Table table, int number) {
    this.table = table;
    this.number = number;
}
Public void run() {
    table.PrintTable(number);
}
}
Public class ThreadSynchronisationExample {

Public static void main (String [] args) {
    Table table = new Table();
    MyThread_1 thread_1 = new MyThread_1(table, 5);
    MyThread_2 thread_2 = new MyThread_2(table, 10);
    thread_1.start();
    thread_2.start();
}
}
```

Ans:- Scripting elements allow you to use Java programming language statements in your JSP pages. They are typically used to create and access objects, define methods, and manage the flow of control. Many tasks that require the use of scripts can be eliminated by using custom tag libraries, in particular the JSP standard Tag library.

Date __/__/__

Scriptlet.jsp

```
<HTML>
<Head><title> Scriptlet </title>
</Head>
<body>
<H3> -- Data Flair -- </H3>
<H3> WL of Scriptlet in JSP </H3>
<%  int a=3;
    int b=4;
    int c=5;
    Out.println("a is: "+a"<br>"+b+"<br>"+
c. is: "+c+"<br>");
    out.println("Multiplication gives: "+a*b*c"<br>");
    out.println("Addition gives: "+(a+b+c));
    %>
</body>
</html>
```

* Scripting elements

1) Expression tags
2) Scriptlet Tags
3) Declaration Tags

1) Expressions:- These Suggest more of evaluate expressions. They incorporate arithmetic & logical expressions so that they can be evaluated. They form an easy means to access the access the value of a java variable

Date _/_/_

2) Scriptlet Tag:- It helps in embedding Java in html for JSP code. <%. — %> tag contains the Java code and are called Scriptlet tags. Scriptlet have a set of statements in Java language. It also allows writing Java code in it.

3) JSP Declarations

⇒ Declaration in JSP declare java language statements <%! — %> tag contain declaration. They can declare class / instance / inner class / variable / methods.