Name : Rashika Khatri

Roll no : 1918594

section : B

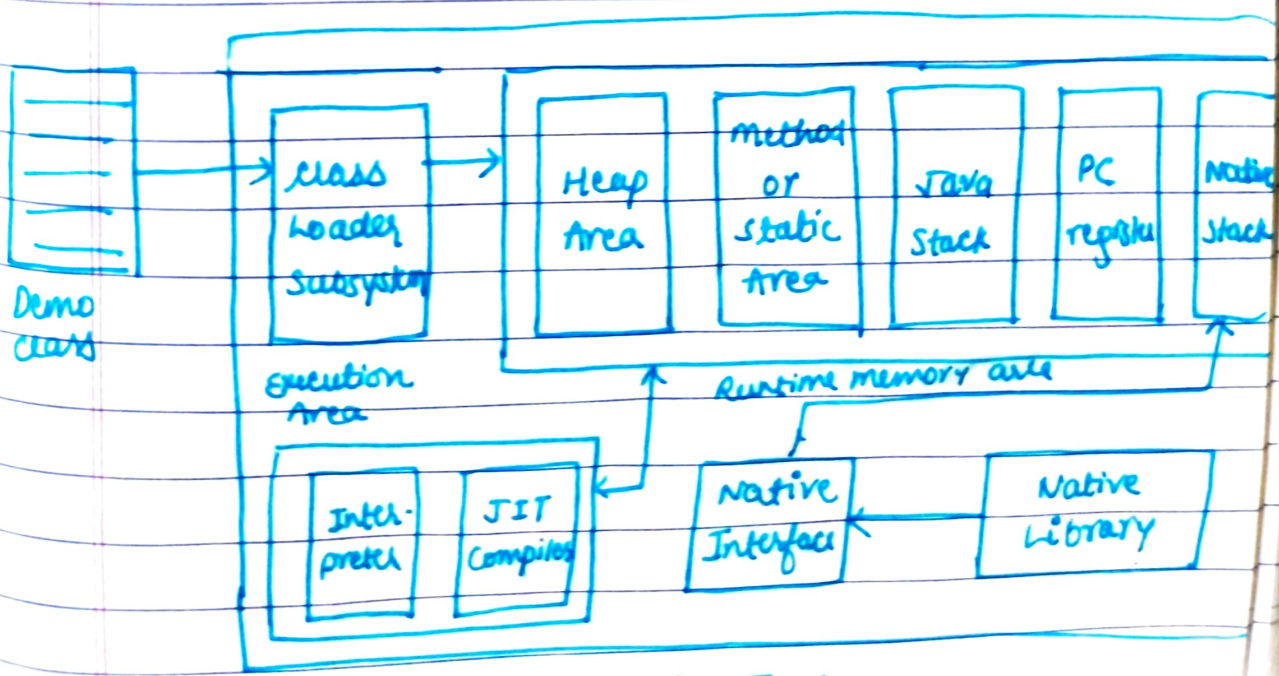
Subject : Java Programming.

Subject Code: TCS 408.

Date.
Page No.

1) a) Java Virtual Machine (JVM) is the heart of entire Java program execution process. First of all the java program is converted into a .class file consisting of ~~by~~ the byte code instructions by the java compiler at the time of compilation. The .class file is given to the JVM and it performs following operations:

- Allocating sufficient memory space for class properties
- Provide runtime environment in which java bytecode can be executed.
- Converting bytecode instructn into machine level instruction.



Internal architecture of JVM

Date.
Page No.

In JVM, there is a module called class loader sub-system which performs following instructions.

→ First of all, it loads the .class file into memory.

→ Then it verifies whether all byte code instructions are proper or not. If it finds any instruction suspicious, then execution is rejected immediately.

→ If the byte instructn are proper, then it allocate necessary memory to execute the program. This memory is divided into 5 parts, called run-time memory areas, which contain the data & result while running the program. These areas are:

1. Heap area: All the objects and its corresponding instance variables and arrays will be stored here.
2. Method area: All the class level data will be stored here including static variables, static method & static block.

Date. _____
Page No. _____

3. Java stack Area: In this all the non-static variable of class is stored.

4. PC Register: It hold the address of current executing instruction once the instruction is ~~completed~~ executed PC register will be updated

5. Native method stack: Java uses native method stack for execution of native methods.

Date. _____
Page No. _____

5) b) There are 5 steps to connect any Java application with the database using JDBC. These are as follows:

Register the driver class
Create connection
Create statements
Execute Queries
Close connection

For Oracle: The driver class for the oracle database is:
"oracle.jdbc.driver.OracleDrivers".

- Connection Url :- "jdbc:oracle:thin:@localhost:1521XE"
  - → @localhost is the server name
  - → 1521 is the port
  - → XE is the oracle service name.

MySQL :-
driver :- "com.mysql.jdbc.Driver"
Connectn Url :- "jdbc:mysql://localhost:3306/sonoo", "root", "root"

Date.
Page No.

Driver Manager class :-

- It acts as an interface b/w user & driver.
- It keep track of the drivers that are available & handles a stable connectn b/w database & driver.

Connection Interface :-

- A connection is the session b/w java application & database.
- The connection interface is the factory of statement, prepared statement & metadata.

Statement Interface :-

Common methods are :-

- public Resultset executeQuery (String sql);
- public int executeUpdate (String sql)
- public boolean execute (String sql);
- public int[] executeBatch();

Eg. to connect Java applicatn to MySQL Database

```
import java.sql*
class MySQLCon{
  public static void main (String args[])
  {
    try {
        class forName ("com.mysql.jdbc.Driver");
      Connection con = DriverManager.getConnection
          ("jdbc:mysql://localhost:3306/sonoo","root"
          ,"root");
  Statement smt = con.createStatement();
  ResultSet rs = smt.executeQuery("select * from emp");
while (rs.next()).
{
    system.out.println(rs.getInt(1)+" "+rs.getString(2);
}
 con.close();
}
Catch (Exception e)
{
    system.out.println(e);
}}
```

Output: 1. RAM
2. SHYAM
3. ROHAN

id	name
1.	RAM
2.	SHYAM
3	ROHAN

Date.
Page No.

2) a) i) Polymorphism is the ability of an object to take on many form.
The most common use of polymorphism in OOP occur when a parent class reference is used to refer to a child class object.

In Java polymorphism is mainly divided into two types:

- Compile time polymorphism
- Run time polymorphism

1. Compile time polymorphism:
It is also known as static polymorphism.
This type of polymorphism is achieved by function overloading or operator overloading.
When there are multiple functions with same name but different parameter then these functions are said to be overloaded.

Date.
Page No.

- 2. Runtime Polymorphism : It is also known as dynamic method dispatch. It is a process in which a function call to the overridden method is resolved at runtime. This type of polymorphism is achieved by method overriding.

Eg:

```
class Parent {
    void Print()
    {
        system.out.print("Parent class");
    }
}
class subclass1 extends Parent
{
    void Print()
    {
        system.out.println("subclass1");
    }
}
class subclass 2 extends Parent
{
    void print() {
    system.out.print("subclass 2");
    }
}
```

Date.
Page No.

```
class Main {
    public static void main (String [] args)
    {
        Parent a = new Parent();
        a.Print();
        a = new subclass1();
        a.Print()
        a = new subClass2();
        a.Print();
    }
}
```

Output :

Parent class
subClass 1
subclass 2.

ii) this keyaoorde :- The this keyword refers to the current object in a method or constructor.

The most common use of this keyword is to eliminate the confusion b/w class attributes and parameters with the same

Date.
Page No.

name. this keyword is used to:
invoke current class constructor.
invoke current class method.
return the current class object
Pass an argument in the method call.

Final keyword :
The final keyword is a non-access modifier used for classes, attributes & methods.
The final keyword is used when you want a variable to always store the same value.
For eg:
PI (3.14159).

Date.
Page No.

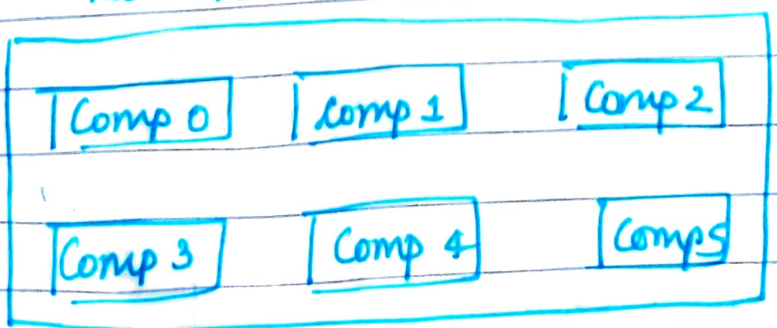
4) a) Java Layout Manager: It is used to position & place components in a container. Layout Manager is an interface that is implemented by all the classes of layout managers. There are 3 basic layout managers which control how UI components are organised on frame:

Flow layout
Grid layout
Border layout.

Once created the layout can be set in the content pane using set layout.

- Flow layout: With flow layout, the components arrange themselves from left to right in order they were added.

Eg: Rows / buttons are left aligned using FlowLayout.LEFT



Date. _____
Page No. _____

Method :-

Set Alignment

FlowLayout LEFT. FlowLayout CENTER.

FlowLayout RIGHT

- Grid Layout : With grid layout, the components arrange themselves in a matrix formation (rows, colums)

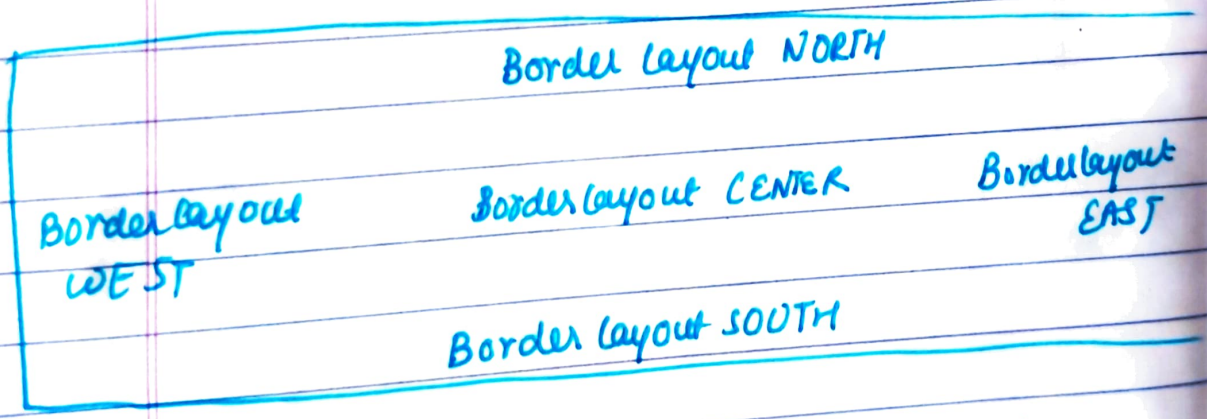
Date.
Page No.

- divides container into grid
- components placed in rows & columns
- The dominating parameter is the row.

Constructors

Grid Layout (row, colums, hGap, V. Gap)

- Border Layout



Components are added to frame using a specified index.

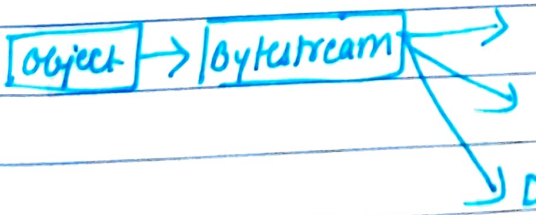
```
Container add (new JButton ("East"). BorderLayout.EAST);
JPanel p = new JPanel (new BorderLayout());
p.setLayout (new BorderLayout ("EAST"));
p.setLayout (new BorderLayout (p, BoxLayout.PAGE_AXIS));
```

Date.
Page No.

Methods :-

- Constructor : BorderLayout (hGap, vGap);
- Adding Components
  mylayout.add (Component, position)
- The default location for a component is BorderLayout CENTER.

8) Object serialization : Serialization is a mechanism of converting the state of an object into a byte stream. Object can be represented as a sequence of bytes that include the object's data as well as information about object's type & type of data stored in object.



Serialization in java is a mechanism. It is mainly used in Hibernate & JMS technologies.

for serializing the object, we call the writeObject() method. ObjectOutputStream

Advantage : It is mainly used to travel object's state on the network.

Entire process is JVM independent, meaning an object can be serialized on one platform & deserialized on an entirely different platform.

Code :-

```
import java.io.*
class Persist {
  public static void main(String args[])
  {
    try
    {
      Student s1 = new Student(211, "Ravi");
      FileOutputStream fout = new FileOutputStream("f.txt");
      ObjectOutputStream out = new ObjectOutputStream(fout);
      out.writeObject(s1);
      out.flush();
      out.close();
      System.out.println("success");
}
```

Date.
Page No.

```
catch (Exception e)
{
    System.out.println(e);
}
}
```

Output ⇒ Success.