Name : Ankit Girdhar Bohare
Roll no : MC - 1 - 20
Class : MCA (1st - Year) DIV - A
Subject : Java Programming [IT - II]

Theory Assignment

1) Discuss the Object Life Time & Garbage Collection.

⇒ In java, the management of memory is handled by the Java Virtual Machine (JVM). The JVM is responsible for allocating and reclaiming memory used by Java objects. The Object Life time and Garbage collection plays a crucial role in Java's Memory Management.

Object Lifetime

* Creation - Objects are created using the 'new' keyword or the other instantiation mechanisms, and the memory is allocated for the object on the heap.

* Reference - The object created is accessible through a reference variable. As long as there is at least one reference pointing to the object, it is considered "alive".

* Usage - The object is actively used in the program, and its methods & properties are accessed through the reference.

* Scope - The object is within the scope as long as there are references to it. It can be used & manipulated.

* Destruction - When there are no more references to the object, it becomes eligible for garbage collection.

Program

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

        // Object Creation
        MyClass obj1 = new MyClass ("Object 1");

        // Object Reference
        MyClass obj2 = obj1;
        // Both obj1 & obj2 refer to the same object.

        // Object Usage
        System.out.println (obj1.getName());
        System.out.println (obj2.getName());

        // Object Destruction
        // When there are no references, the object becomes
           eligible for Garbage Collection
    }
}
```

```
class MyClass {
  private String name;

  public MyClass(String name) {
    this.name = name;
  }
  public String getName() {
     return name;
  }
}
```

Garbage Collection

- Garbage Collection in java is the automated process of deleting code that's no longer needed or used.
- Java uses an automatic garbage collector, which means developers don't need to manually free the memory in languages like C++.
- The garbage collector identifies objects that are no longer reachable or referenced by any part of the program.
- It marks objects that are still in use, creating a "live set" of objects. And then sweeps through the heap and reclaims memory occupied by the objects not marked as live.
- Objects can have a 'finalize()' method that the garbage collector calls before the object is reclaimed.

- However, the relying on 'finalize()' is discouraged, and developers are encouraged to use other methanisms (ie 'try-with-resources').

```
public class Finalization_Ex {
    public static void main(String[] args) {
        MyClass obj = new MyClass("Object with Finalize");

        // Object is still in use
        System.out.println(obj.getName());
        obj = null;
        // Making the reference null, making it eligible
           for Garbage collection
    }
}
class MyClass {
    private String name;
    public MyClass(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
    @Override
    protected void finalize() throws Throwable {
        // Finalization logic
        System.out.println("Finalizing Object: " + name);
        Super.finalize();
    }
}
```

2) Write a Program to override abstract class Method.

⇒ Program:

```
// Abstract class with an abstract method
abstract class Shape {
    // Abstract method
    abstract void draw();
    // Non-abstract method
    void displayArea() {
        System.out.println("Area Calculation");
    }
}
// Concrete class extending the abstract class
class Circle extends Shape {
    private double radius;

    // Constructor
    public Circle(double radius) {
        this.radius = radius;
    }
    // Implementing the abstract method
    @Override
    void draw() {
        System.out.println("Drawing a Circle");
    }
}
```

```
    // Overriding a non-abstract method (optional)
    @Override
    void displayArea() {
        double area = Math.PI * radius * radius;
        System.out.println("Area of Circle: " + area);
    }
}

public class Abstract_Class_Demo {
    public static void main(String[] args) {

        // Creating an object of Circle class
        Circle circle = new Circle(5.0);

        // Calling abstract method
        circle.draw();
        // Calling non-abstract method
        circle.displayArea();
    }
}
```

3) Explain Import and Static import with suitable example.

⇒ Import statement

- It is used to access a class or method from another package we need to use the fully qualified name or we can use import statements.
- The class or method should be accessible. Accessibility is based on the access modifiers.
- Private members are accessible only within the same class. So we don't be able to access a private member even with the fully qualified name or an import statement.
- The java.lang package is automacally imported into our code by Java.

Example:

```
import java.util.Vector;
public class ImportDemo {
    public ImportDemo() {
    // Imported using keyword, hence able to access directly
        Vector v = new Vector();
        v.add("Tea");
        v.add("Coffee");
        v.add("Juice");
        System.out.println("Vector values are: " + v);
```

```
    // Package not imported, hence referring to package
    java.util.ArrayList list = new java.util.ArrayList();
    list.add("Vadapav");
    list.add("Sandwich");
    System.out.println("ArrayList values are: " + list);
  }
  public static void main(String[] args) {
    new ImportDemo();
  }
}
```

Output :

```
Vector values are : [Tea, Coffee, Juice]
ArrayList values are : [Vadapav, Sandwich]
```

Static Import Statement

- Static imports will import all static data so that can use without a class name.
- A static import declaration has two forms, one that imports a particular static member which is known as single static import and one that imports all static members of a class which is known as a static import on demand.
- It was introduced in Java 5 version
- One of the advantages of using static imports is reducing keystrokes and reusability.

Example :

```
import static java.lang.System.*;    // Static Import
public class StaticImportDemo {
    public static void main(String[] args) {
        // System.out is not used as it is imported
        // using the keyword static.
        out.println("Welcome to Goa, Singham");
    }
}
```

Output : Welcome to Goa, Singham

4) Differentiate String Builder and String Buffer with suitable example.

⇒ Java provides three classes to represent a sequence of characters : String, StringBuffer and StringBuilder.
- The String class is an immutable class: whereas StringBuffer & StringBuilder classes are mutable.

StringBuffer	StringBuilder
(1) StringBuffer class is synchronized ie thread safe.	(1) StringBuilder is non-synchronized ie. not thread safe.
(2) It is less efficient than StringBuilder.	(2) It is more efficient than StringBuffer.

(3) It means that two threads can't call the methods of StringBuffer simultaneously.	(3) It means that two threads can call the methods of StringBuilder simultaneously.
(4) StringBuffer was introduced in Java 1.0	(4) StringBuilder was introduced in Java 1.5

StringBuffer Example :

```
public class BufferTest {
    public static void main (String[] args) {
        StringBuffer buffer = new StringBuffer ("Hello");
        buffer.append (" Java");
        System.out.println (buffer);
    }
}
```

Output : HelloJava

StringBuilder Example :

```
public class BuilderTest {
    public static void main (String[] args) {
        StringBuilder builder=new StringBuilder ("Hello");
        builder.append (" Java");
        System.out.println (builder);
    }
}
```

Output : HelloJava

5) Write a program to demonstrate the Inter Thread Communication.

⇒ Inter Thread Communication

```
class Customer {
  int amount = 10000;

  synchronized void withdraw (int amount) {
  System.out.println ("Going to withdraw ... ");

  if (this.amount < amount) {
  System.out.println ("Less balance...");
  try {
     wait();
  } catch (Exception e) {
    }
}
synchronized void deposit (int amount) {
  System.out.println ("Going to deposit ...");
  this.amount += amount;
  System.out.println ("Deposit completed...");
  notify();
  }
}
class Test {
  public static void main (String [] args) {
    final Customer c = new Customer ();
```

```
        new Thread() {
            public void run() {
                c.withdraw(15000);
            }
        }.start();
        new Thread() {
            public void run() {
                c.deposit(10000);
            }
        }.start();
    }
}
```

Output :

```
Going to withdraw..
Less balance...
Going to deposit...
Deposit completed...
Withdraw completed...
```

6) Explain Sequence, Map with suitable example.

⇒ - In Java, 'Sequence' is not ~~used~~ a standalone class or interface in the standard library.
- However, I'll assume you might be referring to a more general Sequence of elements, which can be represented using various data structures like arrays or lists.
- On the other hand, 'Map' refers to the 'java.util.Map' Interface and its implementations, which represent key value pairs.

Sequence (List):

A sequence in java is often represented using a 'List' Interface or one of its implementations like 'ArrayList' or 'Linked List'.

eg.

```
import java.util.ArrayList;
import java.util.List;
public class Sequence {
    public static void main (String[] args) {

    // Creating a List (Sequence) & adding elements to it
    List<String> sequence = new ArrayList<>();
    sequence.add ("Element 1");
    sequence.add ("Element 2");
    Sequence.add ("Element 3");
```

```
    // Accessing elements in the sequence
    System.out.println("Elements in the sequence: ");
    for (String element : sequence) {
        System.out.println(element);
    }
}
}
```

Output:

```
Elements in the sequence:
Element 1
Element 2
Element 3
```

Map

A 'Map' in java represents a collection of key-value pairs, where each key is associated with exactly one value. Common implementations include 'HashMap', 'TreeMap', & 'LinkedHashMap'.

eg.

```
import java.util.HashMap;
import java.util.Map;

public class MapExample {
    public static void main(String[] args) {

        // Creating a Map
        Map<String, Integer> myMap = new HashMap<>();
```

```
// Adding key-value pairs to the Map
myMap.put("One", 1);
myMap.put("Two", 2);
myMap.put("Three", 3);

// Accessing values using keys
System.out.println("Value for key 'Two': "
                   + myMap.get("Two"));

// Iterating over key-value pairs
System.out.println("Key-Value pairs: ");
for (Map.Entry<String, Integer> entry : myMap.entrySet()) {
    System.out.println(entry.getKey() + ": "
                       + entry.getValue());
}
}
}
```

Output :

```
Value for key 'Two' : 2
Key-Value pairs :
One : 1
Two : 2
Three : 3
```

7) Explain adapter class with example.

⇒ Adapter Classes

- Java adapter classes provide the default implementation of listener interfaces.
- If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces.
- The adapter classes are found in java.awt.event, java.awt.dnd & javax.swing.event packages.
- It assists the unrelated classes to work combinely.
- It provides ways to use classes in different ways.
- It increases transparency & reusability of classes.
- It provides a pluggable kit for developing an application.

Adapter Class & their Listener interfaces

WindowAdapter	WindowListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
FocusAdapter	FocusListener
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
HierarchyBoundsAdapter	HierarchyBoundsListener

Example :

```
import java.awt.*;
import java.awt.event.*;
public class AdapterExample {
    // object of Frame
    Frame f;
    // class constructor
    AdapterExample() {
        // Creating a Frame with Title
        f = new Frame("Window Adapter");
        // adding WindowListener to the Frame
        // overriding the windowClosing() method
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                f.dispose();
            }
        });
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new AdapterExample();
    }
}
```

5) Write a Servlet Program to find GCD of two given values. (use web.xml, index.html and Servlet.java)

⇒ index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title> GCD Calculator </title>
</head>
<body>
  <h2> GCD Calculator </h2>
  <form action="gcd" method="post">
    Enter First no: <input type="text" name="num1">
    <br>
    Enter Second no: <input type="text" name="num2">
    <br>
    <input type="submit" value="Calculate GCD">
  </form>
</body>
</html>
```

GCDServlet.java

```
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
```

```
@WebServlet("/gcd")
public class GCDServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
    // Get values from the form
    String n1 = request.getParameter("num1");
    String n2 = request.getParameter("num2");
    try {
        // Parse the values to integers
        int num1 = Integer.parseInt(n1);
        int num2 = Integer.parseInt(n2);
        // Calculate GCD
        int gcd = calculateGCD(num1, num2);

        // Prepare the response
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h2>GCD Result:</h2>");
        out.println("<p>GCD of "+num1+" & "+num2+" is: "+gcd+"</p>");
        out.println("</body></html>");
    } catch (NumberFormatException e) {
        // Handle the case of invalid input
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h2>ERROR</h2>");
        out.println("<p>Please enter valid integers</p>");
        out.println("</body></html>");
    }}
```

```
    private int calculateGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        } return Math.abs(a);
    }
}
```

'web.xml'

```
<?xml version="1.0" encoding="UTF-8" ?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/
           XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns
           /javaee http://xmlns.jcp.org/xml/ns/javaee/
           web-app_4_0.xsd" version="4.0">
  <servlet>
    <servlet-name>GCDServlet</servlet-name>
    <servlet-class>GCDServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>GCDServlet</servlet-name>
    <url-pattern>/gcd</url-pattern
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

19) Write a program to show CRUD operation using JDBC. Assume suitable table structure

⇒ Database (Table : students)

```
create table students (
    id int Primary Key Auto_Increment,
    name varchar(50),
    age int,
    grade varchar(10) );
```

Program :

```
import java.sql.*;

public class JDBC_CRUD {
  public static void main(String[] args) {
    // JDBC Variables for opening, closing & managing connections
    Connection con = null;
    Statement st = null;
    PreparedStatement ps = null;
    try {
      // Open Connection
      con = DriverManager.getConnection("jdbc:mysql://
            localhost:3306/your_database", "your_username",
            "your_password");

      // Execute a query
      st = con.createStatement();
```

```
// CREATE (Insert new record)
String ins = " INSERT INTO students (name, age, grade)
              VALUES (?, ?, ?) ";
ps = con.prepareStatement (ins);
ps.setString (1, "Alice");
ps.setInt (2, 20);
ps.setString (3, "A");
int insR = ps.executeUpdate ();
System.out.println("Record inserted: " + insR);

// READ (select records)
String sel = " SELECT * FROM students ";
ResultSet rs = st.executeQuery (sel);
System.out.println ("Reading records : ");
while (rs.next()) {
   System.out.println (" ID: " + rs.getInt ("id") +
                       ", Name: " + rs.getString ("name") +
                       ", Age: " + rs.getInt ("age") +
                       ", Grade: " + rs.getString ("grade"));
}

// UPDATE (Modify record)
String upd = "UPDATE students SET age=? WHERE name=?";
ps = con.prepareStatement (upd);
ps.setInt (1, 21);
ps.setString (2, " John");
int updR = ps.executeUpdate ();
System.out.println (" Record updated: " + updR);
```

```
// DELETE (Remove record)
String del = "DELETE FROM students WHERE name = ?";
ps = con.preparseStatement(del);
ps.setString(1, "Alice");
int delR = ps.executeUpdate();
System.out.println("Record deleted: " + delR);
} Catch (SQLException e) {
    e.PrintStackTrace();
} finally {
    // close the connection, statement & prepared
    // statement
    try {
        if (st != null) st.close();
        if (ps != null) ps.close();
        if (con != null) con.close();
    } Catch (SQLException e) {
        e.printStackTrace();
    }
  }
 }
}
```

10) Design a Registration Form for Solo Dance Competion using Swing and store the information into the Registration Table. (kindly take 5 different component).

⇒ Program :

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class DanceRegistration extends JFrame {
    private JTextField nameField, agefield, danceStylefield,
            contactfield, emailfield;
    public DanceRegistration() {
        // Set up the frame
        super("Solo Dance Competition Registration Form");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);
        setLocationRelativeTo(null);
        // Create a JPanel with GridLayout
        JPanel panel=new JPanel(new GridLayout(6,2,10,10));
        // Add components to panel
        panel.add(new JLabel("Name:"));
        nameField = new JTextField();
        panel.add(nameField);
```

```
panel.add(new JLabel("Age:"));
ageField = new JTextField();
panel.add(ageField);

panel.add(new JLabel("Dance Style:"));
danceStyleField = new JTextField();
panel.add(danceStyleField);

panel.add(new JLabel("Contact no:"));
contactField = new JTextField.();
panel.add(contactField);

panel.add(new JLabel("Email:"));
emailField = new JTextField.();
panel.add(emailField);

JButton submitButton = new JButton("Submit");
submitButton.addActionListener(new ActionListener() {
    storeRegistrationInfo();
@Override
    public void actionPerformed(ActionEvent e) {
        storeRegistrationInfo();
    }
});
panel.add(submitButton);
// add Panel to the JFrame
add(Panel);
// set the JFrame visible
setVisible(true);
}
```

```
private void storeRegistrationInfo() {
    // Get values from fields
    String name = nameField.getText();
    String age = ageField.getText();
    String danceStyle = danceStyleField.getText();
    String contact = ContactField.getText();
    String email = emailField.getText();

    // Validate Form Fields
    if (name.isEmpty() || age.isEmpty() ||
        danceStyle.isEmpty() || contact.isEmpty() ||
        email.isEmpty()) {
            }
        JOptionPane.showMessageDialog(this, "Please
            fill in all fields", "Error", JOptionPane.ERROR_
            MESSAGE);
        return;
    }

    // Store information in Table
    try {
        Connection con = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/your_database",
            "your-username", "your_password");
        String sql = "INSERT INTO Registration (name,
            age, dance, style, contact, email) VALUES
            (?, ?, ?, ?, ?)";
        try (PreparedStatement ps = con.prepareStatement(sql))
        { } {
```

```
            ps.setString(1, "name");
            ps.setString(2, age);
            ps.setString(3, danceStyle);
            ps.setString(4, contanct);
            ps.setString(5, email);
            ps.executeUpdate();

            // Display Success Message
            JOptionPane.showMessageDialog(this, "Registration
                successful!", "Success", JOptionPane.
                INFORMATION_MESSAGE);
        }
    } catch(SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error
            storing registration Information.", "Error",
            JOptionPane.ERROR_MESSAGE);
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new
        DanceRegistration());
    }
}
```