**Instructions Sheet**

**1. Write all answers in the answer booklet.**

**2. This examination comprises two parts - Part A and Part B.**

**3. Part A is worth 20 marks. Students must answer all questions in this part.**

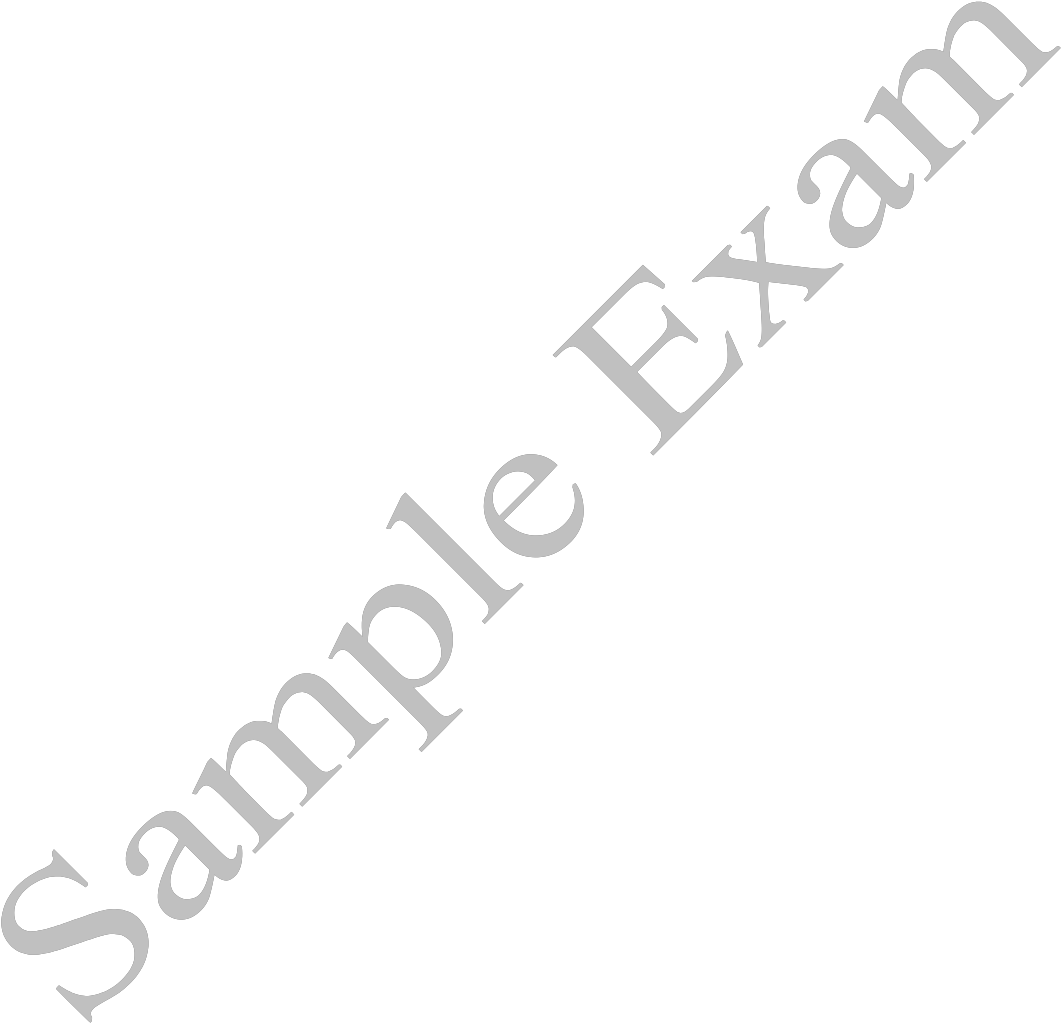
**4. Part B is worth 45 marks. Students can answer any nine (9) questions in this part.**

**PART A 20 MARKS COMPULSORY QUESTIONS**

**Answer ALL questions from this part.**

**Each question is worth 2.0 marks (10 x 2 = 20 marks).**

Briefly comment on the following statements from Java language point of view. For each question, limit your answers to half a page only.

**Question 1 2 Marks**

Abstract class vs concrete class.

**Question 2 2 Marks**

Java language allows implementation of multiple interfaces by one class.

**Question 3 to Question 10 are similar to the above.**

**PART B 45 MARKS COMPULSORY QUESTIONS**

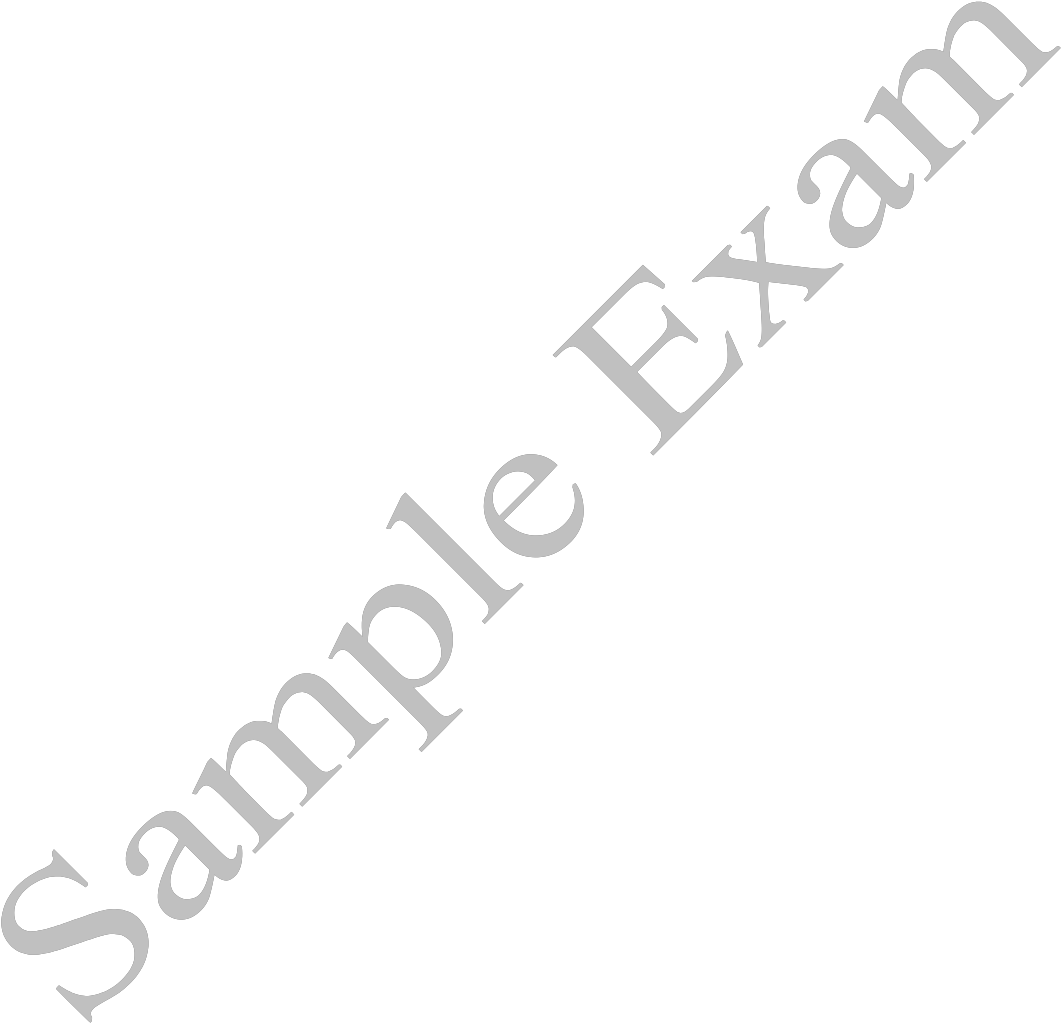
**Answer only nine (9) questions from this part.**

**Question 1 4 MARKS**

The following codes define **Person** class in a file named Person.java. public class Person

{

private String name;

private int age;

public Person(String n, int a)

{

name = n;

age = a;

}

@override

public String toString()

{

return name + “ “ +age;

}

} //end class definition

Your tasks are:

 to define two *set* methods to set the name and age of Person.

 to define two *get* methods to retrieve the name and age of Person.

 to define a method named *isAdult* so that it will return true if the person’s age is

greater than 18.

 to create PersonTester.java file wherein PersonTester class helps you

o to create a Person object with the name value of “Johnson” and the age value of 35.

o to use toString method to display the above mentioned Person object’s

field values

**( Assume PersonTestor.java and Person.java are two separate files )**

**Question 2 8 MARKS**

The following Java code demonstrates the concept of inheritance, where two subclasses **SalaryEmployee** and **HourlyEmployee** inherit the properties and methods from the superclass **Employee**.

Your tasks are:

• Write the missing implementation codes for constructors of SalaryEmployee and

HourlyEmployee classes.

• What would have been the output produced by the program

*DemoInheritance* if it is executed correctly?

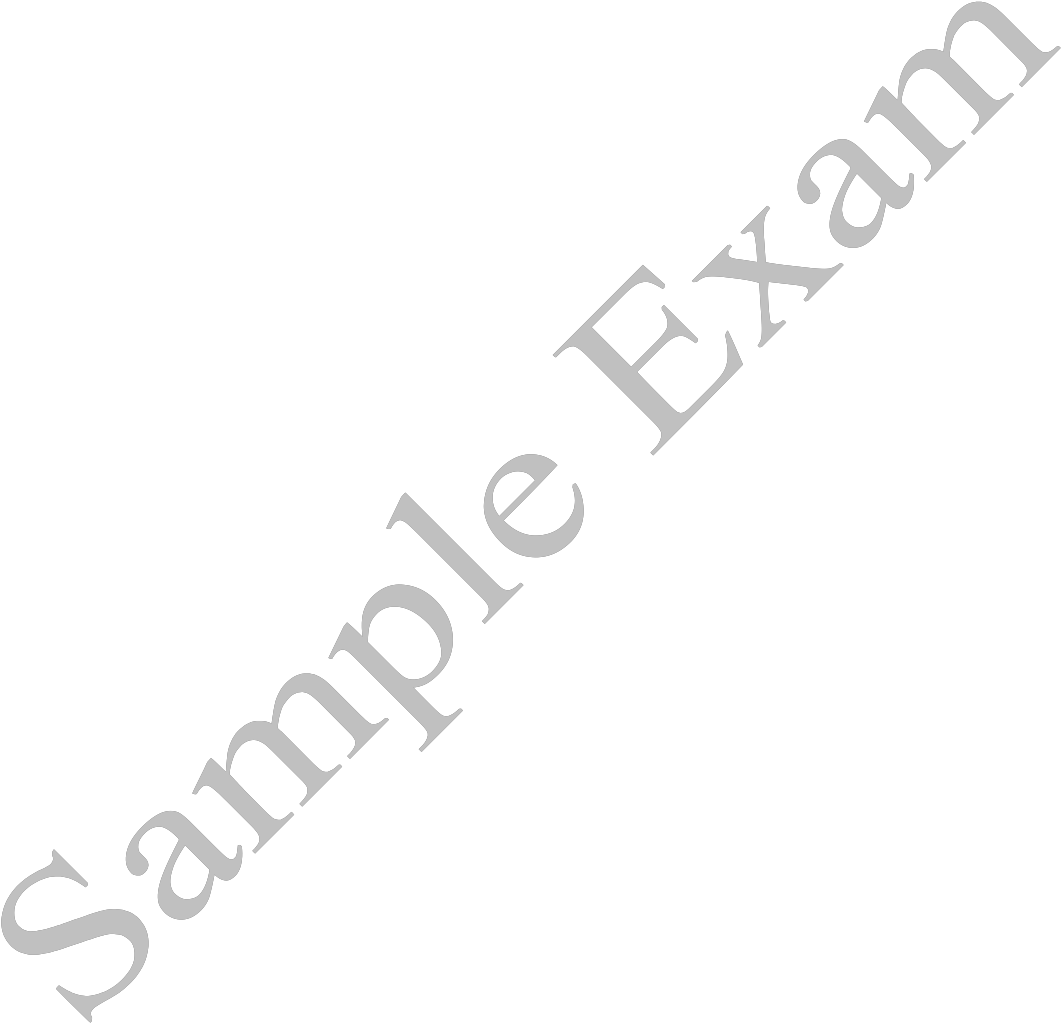
import java.text.DecimalFormat;

class Employee

{

protected String empName;

protected String empSSN;



public Employee(String empName, String empSSN)

{

this.empName = empName;

this.empSSN = empSSN;

}

// update the employee name

public void setName(String empName)

{ this.empName = empName; }

// returns a formatted string to display employee information public String toString()

{ return "Name: " + empName + '\n' + "SS#: " + empSSN; }

} //end of class Employee

class SalaryEmployee extends Employee

{

// new attributes that extends attributes in Employee

private double salary;

public SalaryEmployee(String empName, String empSSN, double salary)

{

// missing lines of codes

}

// accessor method to return the salary public double getSalary()

{ return salary; }

// mutator method to update the salary public void setSalary(double sal)

{ salary = sal; }

// return a formated string with salaried employee information

// including name, ssn, status (salaried) and monthly pay

public String toString()

{

DecimalFormat fmt = new DecimalFormat("#.00");

return super.toString() + '\n' +

"Status: Salary" + '\n' +

"Salary: $" + fmt.format(salary) + "\n";

}

}//End of class SalaryEmployee

class HourlyEmployee extends Employee

{

// specialattributes for hourly pay

private double hourlyPay;

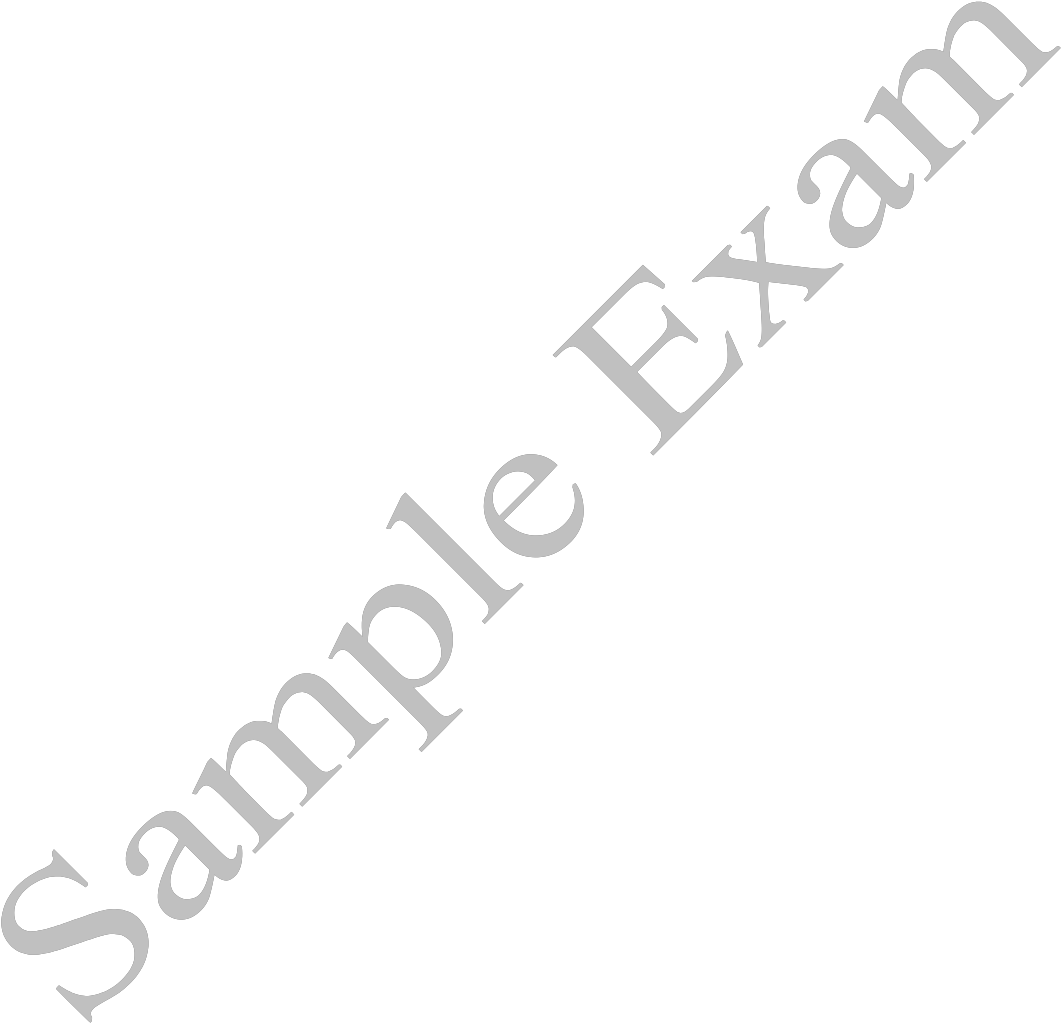
private double hoursWorked;

public HourlyEmployee(String empName, String empSSN, double hourlyPay, double hoursWorked)

{

// missing lines of codes

}

// access and update the hourly pay and hours worked public void setHourlyPay(double hourlyPay)

{ this.hourlyPay = hourlyPay; }

public void setHoursWorked(double hoursWorked)

{ this.hoursWorked = hoursWorked; }

// access and update the hourly pay and hours worked public double getHourlyPay()

{ return hourlyPay; }

public double getHoursWorked()

{ return hoursWorked; }

// call toString() from superclass and add info type of

// employee (hourly), hourly pay rate and hours worked

public String toString()

{

DecimalFormat fmt = new DecimalFormat("#.00");

return super.toString() + '\n' +

"Status: Hourly" + '\n' +

"Rate:$" + fmt.format(hourlyPay) + "\n" +

"Hours: " + fmt.format(hoursWorked) + "\n";

}

}// End of HourlyEmployee public class DemoInheritance

{

public static void main(String[] args)

{

HourlyEmployee hEmp = new HourlyEmployee("Steve Howard", "896-54-

3217",10.50,40);

SalaryEmployee sEmp = new SalaryEmployee("Moira Dunn", "456-14-

3787",800.0);

System.out.println(hEmp); System.out.println(sEmp);

}

}

**Question 3 6 MARKS**

Draw UML diagram with respect to the class Employee and its subclasses which have been mentioned in the above question number 2.

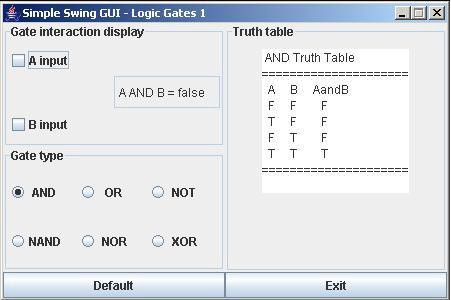
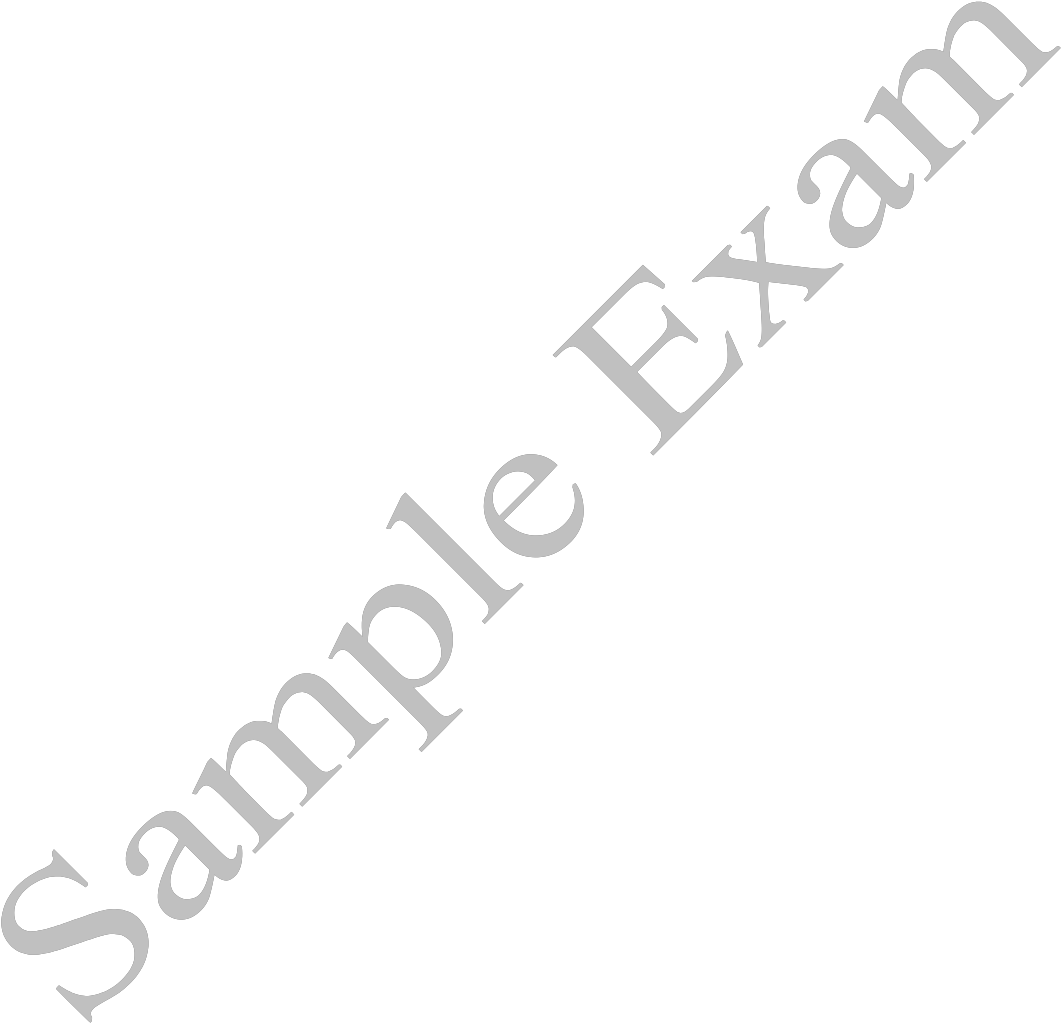
**Question 4 6 MARKS**

Identify the javax.swing components from the following GUI. Explain how the layout managers can be used for this GUI.

(Note: The javax.swing components must be written by using correct upper case and

lower case letters. i.e., same class names as defined in javax.swing)

**Question 5 6 MARKS**



Insert the missing codes in the following Java program.

//Class PlayListener is an action listener import javax.swing.JButton;

import javax.swing.JFrame;

**//Complete the two missing import statements below:**

import java.awt.event.

Missing code ;

import java.awt.event. ;

Missing code

class PlayListener implements ActionListener

{

public void actionPerformed(ActionEvent event)

{

System.out.println("Play is selected");

}

}

// Class PlayTester is a GUI to use PlayListener

public class PlayTester extends JFrame

{

JButton bt;

PlayTester()

{

JButton bt = new JButton("Start Play");

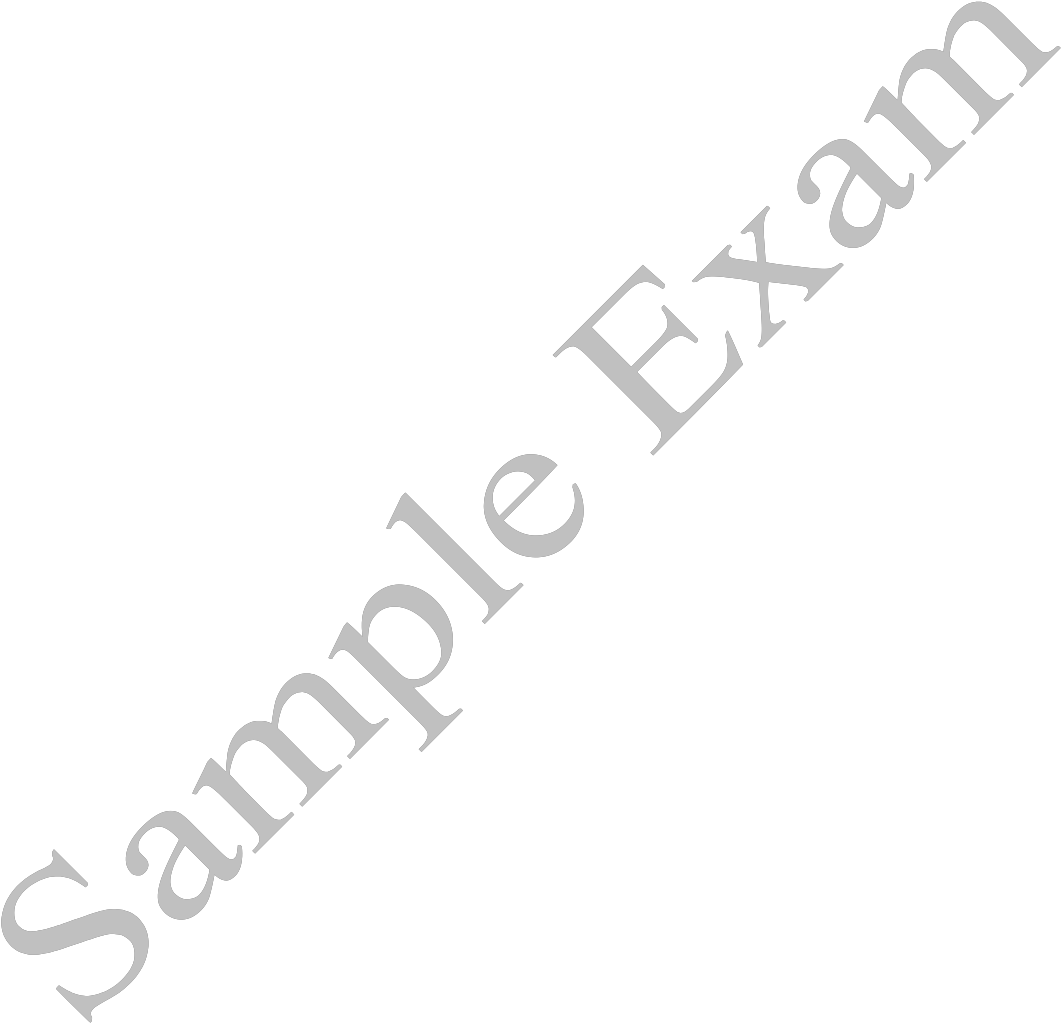
add(bt);

**//Write your code here to install PlayListner**

**//Write your code related to ‘Start Play’ button-click**

**//event handled by the PlayListener**

Missing lines of codes

}

public static void main(String [] args)

{

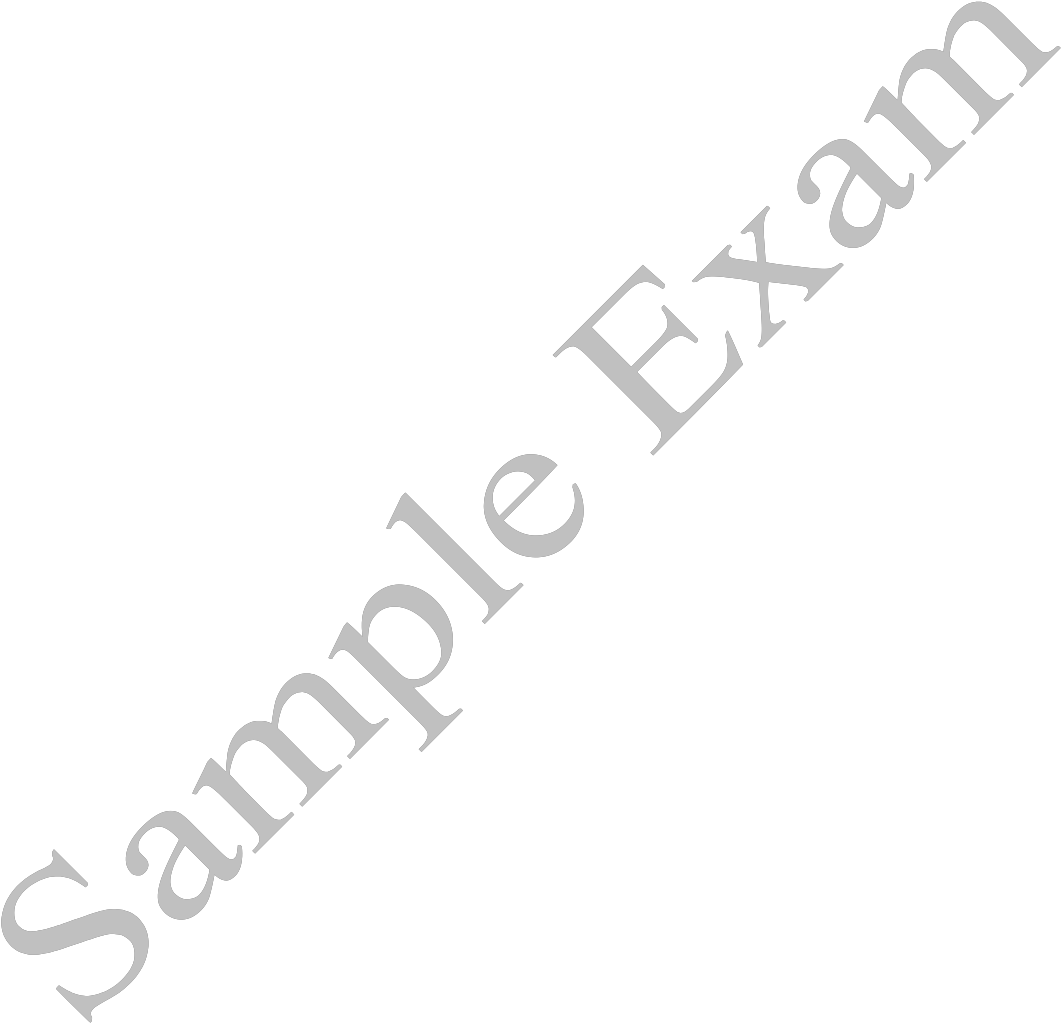
PlayTester aGame = new PlayTester(); aGame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); aGame.setSize(300, 300);

aGame.setVisible(true);

}

}

**Question 6 5 MARKS**

Assume that the following program compiles and runs without any errors. What is the output produced by this program.

The third ‘for’ loop has to be rewritten by using an iterator. The iterator should iterate through the teamList object, retrieve the data from teamList and display the retrieved data on screen.

import java.util.LinkedList;

public class MyProg

{

public static void main(String[] args)

{

String[] team = {"Australia","New Zealand",

"India","UK","Pakistan","China"};

LinkedList<String> teamList = new LinkedList<String>();

for (int i=0; i<team.length; i++)

{

teamList.add(team[i]);

}

for (int i=0; i<teamList.size(); i++)

{

String tempTeam = teamList.get(i);

if (tempTeam.equals("New Zealand"))

{

teamList.remove(i);

}

}

for (int j=0; j<teamList.size(); j++)

{

System.out.println(teamList.get(j).toString());

}

}

}//end of class definition

**Question 7 5 MARKS**

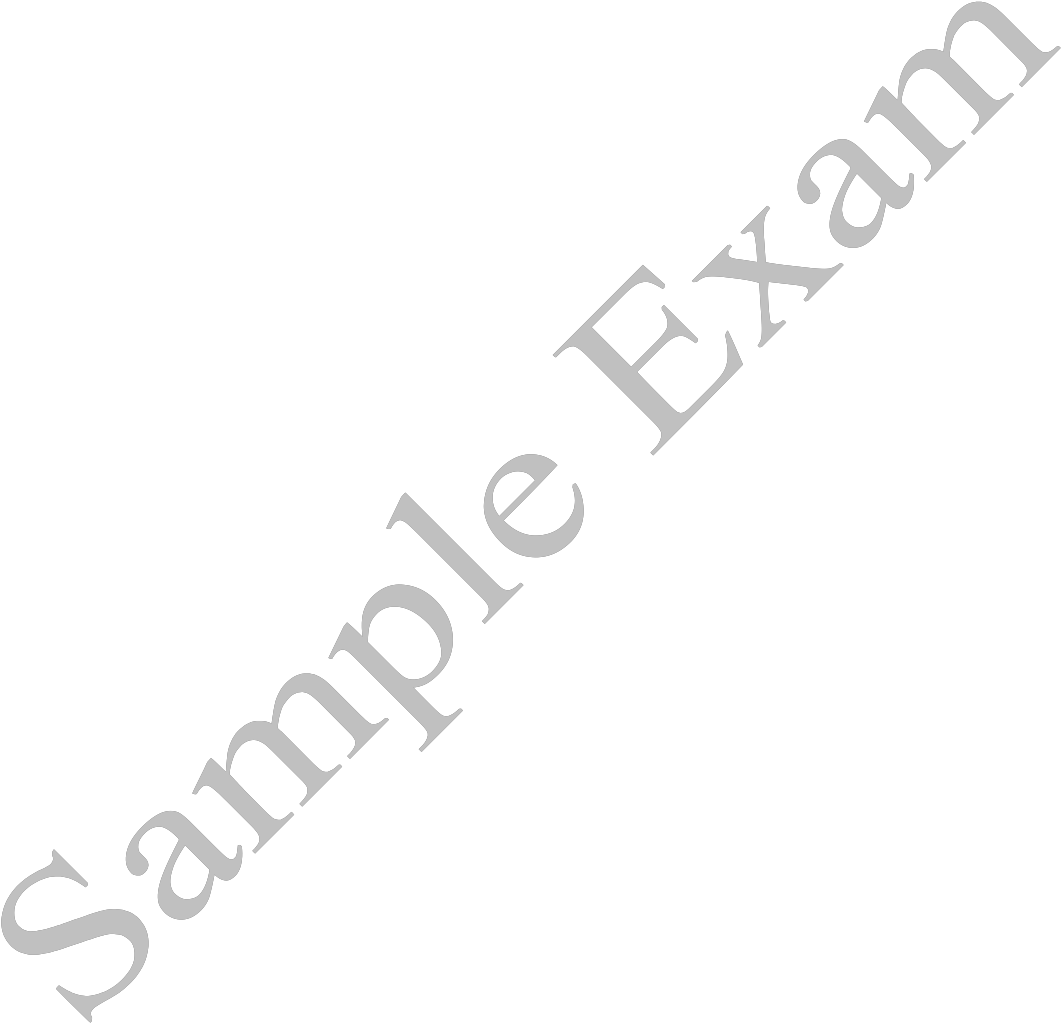
Write a program that uses a Stack object to determine whether a string is a Palindrome or not (i.e., a Palindrome is spelled identically backward and forward. E.g “mum”).

**Question 8 5 MARKS**

(a) Find out whether the tree in the following diagram is a binary search tree or not. Justify your answer. Write down the sequence in terms of the inorder, preorder and postorder traversals of this tree. (**2.5 MARKS**)

40

21 65



8 31 52 77

(b) Given a Big-O estimate for the running time ***T* (*n*) = *n*2 + 1 + *n* log *n ,***

**2**

is this algorithm faster than O(*nlog2n*)? Explain your answer. (**2.5 MARKS**)

**Question 9 5 MARKS**

**……………………..**

**Question 8 5 MARKS**

…………………….

**End of Paper**