#### Question 1 (40 Marks)

1. Briefly describe the use of the static modifier in the context of Java programming, both when applied to variables and when applied to methods.

(6 Marks)

1. Outline four differences between interfaces and abstract classes in Java.

(12 Marks)

1. The abstract class MoneyDevice represents various machines which accept money. One such item represented below is a Jukebox. Trace through the following code and explain the meaning of the following:
   * 1. an abstract class
     2. an abstract method
     3. the protected access modifier

(9 Marks)

abstract class MoneyDevice {

protected int volume;

private Boolean on;

public void switchOn() {

on = true;

}

public void switchOff() {

on = false;

}

public abstract void changeVolume(int volume);

}

class JUKEBOX extends MoneyDevice {

private int songNumber;

public void changeVolume(int amount) {

volume += amount;

}

}

}

1. Write a method as part of the class JUKEBOX to enable the song number to be changed.

(3 Marks)

1. We now want to represent the idea of a MoneyManager that has responsibility of maintaining a record of how much money is collected by such machines.

Write the code required to implement the MoneyManager interface with the following methods:

1. setStartAmount: to set the amount of money collected at the start of day to 0.
2. getDailyTotal: to return the total amount of money collected by the Jukebox in one day.

(8 Marks)

1. Re-write the heading of the JUKEBOX class to incorporate the use of an interface called MoneyManager

(2 Marks)

#### Question 2 (30 Marks)

Consider the following code extract from the definition of a class Building with four members. This class contains an inner member class called Room with two member variables. The completed test class is also provided.

import java.util.ArrayList;

public class Building {

private int sqrFeet;

private String name;

private double cost;

private ArrayList<Room> rList;

public Building() {

this.sqrFeet = 0;

this.name = "";

this.cost = 0.0;

}

public int calculatremainingCapacity() {

return 0;

}

class Room {

private int roomNumber;

private int capacity;

Room(int roomNumber, int capacity) {

this.roomNumber = roomNumber;

this.capacity = capacity;

}

}

}

public class TestBuilding {

public static void main(String argsp[])

{

int[] numbers = {1,2,3};

int[] cap = {300, 500, 600};

Building b = new Building(15000,"Icon Court", 25000.00, numbers, cap);

System.out.println("Remaining capacity in the building is:" +

b.calculateRemainingCapacity());

}

}

1. Write the code for the overloaded constructor in the outer class based on the call to this constructor in the test class. Inside this constructor the member variables should be initialized and the inner class objects should be created and stored in the array list.

(8 Marks)

1. Write the code for the method **calculateRemainingCapacity()** which should calculate and return the remaining capacity in the building after creating the three rooms of varying sizes.

(6 Marks)

1. The code extract provided demonstrates the user of an **Inner Class** and **Composition.** Explain what you understand by these two concepts in Java.

(8 Marks)

1. Explain with the aid of an example the use of a **local class** in Java.

(8 Marks)

#### Question 3 (30 Marks)

1. Within Java it is possible to create *your* *own* exceptions. Discuss why this is desirable and outline how this is achieved.

(10 Marks)

1. The following code consists of 2 classes: an **Account** class and a **TestAccount** class for an Internet banking system.

class Account

{

private double balance;

public Account (double balance)

{

this.balance =balance;

}

public void withdraw(double amt)

{

balance -= amt;

}

public double getBalance()

{

return balance;

}

}

public class TestAccount

{

public static void main(String[] arg)

{

Account theAccount = new Account(200);

theAccount.withdraw(-300);

}

}

1. Develop an exception class **PositiveWithdrawalException** that prints an error message "Withdrawal amount must be positive".

(5 Marks)

1. Modify the above code such that: If a negative amount is entered, a **PositiveWithdrawalException** exception is ***thrown*** within the withdrawal method and is ***caught*** in the calling method (main).

(9 Marks)

1. Modify the above code (again the original code as specified above) such that: If a negative amount is entered, the **PositiveWithdrawalException** exception is handled *locally* within the withdrawal method.

(6 Marks)

#### Question 4 (30 Marks)

This question has been removed as the topic is no longer on the syllabus.