CS&IT

Java

Boost

RMIT School of Computer Science and Information Technology
Introduction

The Java Boost session is designed to assist you in determining the level of Java skills are and assess your skill level in comparison to those expected by lecturers. For returning students they are designed to help refresh previous skills so you can start the new year successfully.

There are lab assistants on hand to help you through any small problem that you may encounter. They can show you how to compile programs, where to find the Java API, and even detect small bugs that you may have overlooked.

If you do encounter any difficulties completing this material, you can talk to your lab assistant.

Solutions to questions are given at the back of this booklet.

Check your work and if you cannot understand the given solution, ask for assistance from your lab assistant.

If you feel unable to attempt a question, it might be useful to look at the solution. Read through the answer carefully. Highlight any parts that you do not understand. Use lecture notes, textbook or online resources, so that you are able to understand the solution.

Wait a day as this can be useful for your brain to process the understandings you have gained from the background work, then attempt the question yourself. Do not look at the solution again until you have completed your own answer. You will learn by doing the problem, not by just reading the answer.
Part I  Multiple Choice / Fill in the blanks
Select the most appropriate response and circle the corresponding letter (A/B/C/D). There is only one correct answer for each question.

1. Which of the following statements is \textbf{not} true in Java?
   A) A long variable can be assigned a long value  
   B) A long variable can be assigned an int value  
   C) A long variable can be assigned a double value  
   D) A long variable can be assigned a byte value  

2. What will be the value assigned to the variable \(x\) as a result of the statement below?
   \[
   \text{int } x = 6 + 12 / 2 * 3
   \]
   A) 3  
   B) 9  
   C) 27  
   D) 24

3. What will be the combined effect of the 3 statements below?
   \[
   \begin{align*}
   &x = x + y; \\
   &y = x - y; \\
   &x = x - y;
   \end{align*}
   \]
   A) To assign \(x\) and \(y\) to the smallest value  
   B) To set both \(x\) and \(y\) to the difference between them  
   C) To swap the values of \(x\) and \(y\)  
   D) None of the above

4. For which values of \(x\) (an int variable), will the program segment below print the message purple?
   
   ```java
   if ( x >= 6)
      System.out.println("red");
   else if ( x < 2)
      System.out.println("blue");
   else
      System.out.println("purple");
   ```
   A) for all values less than 6  
   B) for all values greater than or equal to 2  
   C) for all values less than 6 or greater than or equal to 2  
   D) for all values less than 6 and greater than or equal to 2

5. Consider the for loop:
   
   ```java
   for (int x=1; x<=10; x++)
      System.out.println(x);
   ```
Which of the following while loops is equivalent to the for loop above?

A)  
```java
int x = 1;
while (x <= 10)
    System.out.print(x++);
```

B)  
```java
int x = 1;
while (x++ <= 10)
    System.out.print(x);
```

C)  
```java
int x = 1;
while (++x <= 10)
    System.out.print(x);
```

D)  
```java
int x = 1;
while (x <= 10)
    System.out.print(++x);
```

6. What will be the output of the program below?

```java
public class Test1 {
    public static void main(String args[]) {
        int x = 2;
        int y = f2(f1(x)) - f1(f2(x));
        System.out.println(y);
    }
    public static int f1(int x) {
        return x * 3;
    }
    public static int f2(int x) {
        return x + 3;
    }
}
```

A) 6  
B) 0  
C) -6  
D) None of the above
7. What will be the output of the following program segment?

```java
for (int i=1; i<=5; i++)
{
    if ( i == 3 )
        continue;
    if (i == 4)
        break;
    System.out.print("*");
}
System.out.println("end");
```

A) *end
B) **end
C) ***end
D) ****end

8. How many times will the while loop body below be executed?

```java
int nums[] = {1,5,8,0,3,6,0,9};
int i=0;
while ( nums[i] != 0 ) {
    System.out.println(nums[i]);
    i++;
}
```

A) 3 times
B) 4 times
C) 7 times
D) 8 times

9. Which one of the following statements is false?

A) A throws clause allows a method to propagate an exception
B) There can be at most one catch clause attached to a try block
C) Statements in the finally clause will be executed regardless of whether an exception is thrown and not caught, thrown and caught, or not thrown at all
D) There can be multiple throw clauses in a method

10. Which of the following is a valid declaration of an int array?

A) int a = new int[3];
B) int a[] = new int[];
C) int a[] = new int[3];
D) int a[3] = new int[];
11. What is the output of the program below?

```java
public class Test2DArray {
    public static void main (String[] args){
        int[][] m = new int[3][3];
        int x = 0;
        for (int i=0; i<3; i++)
            for (int j=0; j<3; j++)
                m[i][j] = ++x;
        for (int i=0; i<3; i++)
            for (int j=0; j<3; j++)
                System.out.print("   "+m[i][j]);
        System.out.println();
    }
}
```

A) 1 2 3
   1 2 3
   1 2 3
B) 1 1 1
   2 2 2
   3 3 3
C) 1 2 3
   4 5 6
   7 8 9
D) 1 4 7
   2 5 8
   3 6 9

12. Which of the following is not a valid identifier?

A) grade8Money
B) 8thgradeMoney
C) grade_8_Money
D) $grade8

13. Which of the following parts of a class should always be made private?

A) instance variables
B) accessors
C) mutators
D) constructors
14. In the program below which of the statements within `main()` will result in a compilation error (if any).

```java
class A {}
class B extends A {}
class C extends A {}
public class TestClass {
    public static void main (String[] args) {
        A a1 = new A(); // statement 1
        A a2 = new B(); // statement 2
        A a3 = new C(); // statement 3
        B b = new C(); // statement 4
    }
}
```

A) Statement 1  
B) Statement 2  
C) Statement 3  
D) Statement 4

15. Which one of the following statements (I,II) below is/are true (if any)?

I  An abstract class cannot have constructors  
II  An abstract class cannot be instantiated  
III  Abstract classes must contain abstract methods

A) I only  
B) II only  
C) III only  
D) All of them

16. In the program below, which of the following statements will result in a compilation error?

```java
public class TestScope {
    public static void main(String args[]) {
        int x = 10;
        for (int y=1; y<20; ++y) {
            x++;          // statement 1
            x = x + y;    // statement 2
        }
        int a = x;     // statement 3
        int b = y;     // statement 4
        System.out.println("a = " + a + " b = " + b);
    }
}
```

A) Statements labelled 1, 2, 3 and 4  
B) Statements labelled 2, 3 and 4  
C) Statements labelled 3 and 4  
D) Statement labelled 4
17. What will be the output of the program below.

class A
{  public void method1()
    {  System.out.print(" A1");
    }
  public void method2()
    {  method1();
        System.out.print(" A2");
    }
}

class B extends A
{  public void method2()
    {  super.method2();
        System.out.print(" B2");
    }
  public void method1()
    {  System.out.print(" B1");
    }
}

public class Poly1
{  public static void main(String args[])
    {  B b = new B();
        b.method2();
    }
}

A) A1 A2 B2
B) B1 A2 B2
C) A1 B2
D) A2 B2

18. Which of the following statements about a constructor is false?

A) A constructor must take the same names as the class  
B) A constructor can return a value  
C) A constructor can take any number of arguments  
D) A subclass constructor can call its superclass constructor using super(…)

19. Which of the following loop bodies will be executed at least once?

A) for loop  
B) while loop  
C) do - while loop  
D) nested - loop
In the code segment below assume that methods 1 and 2 will print the messages A and B respectively unless they encounter an error in which case they throw exceptions of type Exception1, Exception2 or Exception3 as shown below. These classes are not related by inheritance. Note that catch clauses are provided for Exception1 and Exception2.

```
try
{  meth1();  // prints A unless Exception1 raised
    meth2();  // prints B unless Exception2 or Exception3 raised
}
catch (Exception1 e)  // type thrown by meth1
{  ...
    System.out.println("C");
}
catch (Exception2 e)  // type thrown by meth2
{  ...
    System.out.println("D");
}
finally
{  System.out.println("E");
}  
System.out.println("F");
```

Which of the following is not a possible sequence of message output?

A)  A B E F
B)  A D E F
C)  A C D E F
D)  A E
Part II (Program segments)
Write your answers in the space provided

21. Write a code fragment using an appropriate loop to print all the even numbers between 1002 and 1020, as shown below. (5 marks)

| 1002 | 1004 | 1006 | 1008 | 1010 | ... | 1020 |

22. Complete the code fragment below to compute the fine payable when a car is caught for speeding (driving above the speed limit). (5 marks)

<table>
<thead>
<tr>
<th>Speed</th>
<th>fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Above Speed Limit by 10km/h or less</td>
<td>$120</td>
</tr>
<tr>
<td>Above Speed Limit by 11 to 30 km/h (inclusive)</td>
<td>$250</td>
</tr>
<tr>
<td>Above Speed Limit by 31 km/h or more</td>
<td>$500</td>
</tr>
</tbody>
</table>

For example, if a car is caught for driving at 75 km/h in a 60 km/h zone the fine is $250, as the actual speed exceeds the speed limit by 15 km/h.

Scanner sc = new Scanner(System.in);
System.out.println("Zone Speed Limit ");
int zoneSpeed = sc.nextInt();
System.out.println("Speed recorded ");
int carSpeed = sc.nextInt();

// now write the code to compute and display the fine
23. Use a nested for loop to print the pattern of stars and hyphens (-) in the form below. Note the sequence in each row consists of a sequence of hyphens followed by sequence of stars and then a sequence of hyphens. Note the number of consecutive stars in each row increases from 1 to 9 in steps of 2, while the number of consecutive hyphens per row decreases from 5 down to 1 in steps of 1. (You are required to write only the code fragment.)

----*----
-----****---
----*******--
----********---
----************--

(5marks)

24. Student marks are stored in the array m (unspecified size). Complete the missing part of the program below to find the highest mark. (5 marks)

** Hint: You can use m.length to get the length of the array referred by m. **

```java
public class FindHighest {
    public static void main(String[] args) {
        int m[] = {28, 67, 94 ... };
        int highest;
        // Write the code here

        System.out.println("Highest mark = " + highest);
    }
}
```
25. You are given a robot object similar to the one used in assignment 2. The methods `extend()`, `contract()` of Robot class extends and contracts the 2\textsuperscript{nd} arm by 1 unit. Similarly the methods `lower()` and `raise()` lower and raise the 3\textsuperscript{rd} arm by 1 unit. The methods `pick()` and `drop()` pick and drop the block that is just underneath the 3\textsuperscript{rd} arm. The initial states of the robot and the blocks just prior to executing the code fragment below are given in the diagram on the left. Sketch the final state of the robot and the blocks in the empty box on the right (5 marks)

```
int ht = 4;
int wd = 4;
for (int i=1; i<=3; i++)
{
    for (int j=1; j<=wd; j++)
        r.extend();
    for (int j=1; j<=4; j++)
        r.lower();
    r.pick();
    for (int j=1; j<=4; j++)
        r.raise();
    for (int j=1; j<=wd; j++)
        r.contract();
    for (int j=1; j<=ht; j++)
        r.lower();
    r.drop();
    ht--;
    wd--;
}
```
26. The abstract class given below allows all fresh students in the school of Computer Science and Information Technology (CS and IT) to enrol up to a maximum of four courses from "Java01", "IntroProg", "DB01", "OS01", "HW01" and "Math". It also allows withdrawal from an enrolled course. You are required to extend this abstract class to meet specific requirements for enrolment of Computer Science (CS) and Information Technology (IT) students (see next page).

(10 marks)

abstract class Student
{
    private String name;
    private String number;
    protected String courses[];  // enrolled courses
    protected int count = 0;  // number of courses enrolled
    private static String current[] =
        {"Java01","IntroProg","DB01","OS01","HW01","Math"};
    private static final int MAX = 4;

    public Student(String name, String number)
    {
        courses = new String[MAX];
        this.name = name;
        this.number = number;
    }

    public int getIndex(String course)
    {
        for (int i=0; i<count; i++)
            if (courses[i].compareTo(course) == 0)
                return i;
        return -1;
    }

    public void enrol(String course) throws Exception
    {
        if (count == MAX)
            throw new Exception("Cannot enrol. Max courses exceeded");
        if (getIndex(course) >= 0)
            throw new Exception(course + " already enrolled");
        for (int i=0; i<current.length; i++)
            if (current[i].compareTo(course) == 0)
            {
                courses[count++] = course;
                return;
            }
        throw new Exception("No such course " + course);
    }

    public void print()
    {
        System.out.println("student name = "+name);
        System.out.println("student number = "+number);
        for (int i=0; i<count; i++)
            System.out.println(courses[i]);
    }

    public void withdraw(String course) throws Exception
    {
        int i;
        if ( (i = getIndex(course)) >= 0)
        {
            for (int j=i; j<count-1; j++)
                courses[j] = courses[j+1];
            count--;
        }
        else throw new Exception("Cannot withdraw " + course);
    }
}
Requirements for subclasses

The subclass for CS Students must ensure that all CS students are enrolled in Java01 before enrolling in any other courses. Similarly it must not allow CS students to withdraw Java01 until all other courses are withdrawn. In other words, if there is only one course CS students are enrolled in it must be Java01.

The subclass for IT students must ensure no IT student is enrolled in Java01 and IntroProg at the same time.

Your code must throw Exceptions when necessary with appropriate (explicit) error messages. You are not allowed to change the super class Student in any way.

class CSStudent …
class ITStudent ...
Part III - Program writing
Write your answers in the space provided

28. A simple application for billing patients in a medical clinic

Overview

You are required to write a (daily) billing program for a medical clinic. This program must compute and produce a summary of consultations and other related fees charged by specialist and generalist doctors (medical practitioners).

The main intention of this section is to allow you to demonstrate your understanding of the concepts of encapsulation, inheritance, polymorphism, abstract classes, exception-handling and file writing. You are encouraged to add comments to explain your program design.

Where the specification is subject to interpretation, you may make any reasonable assumptions but you are required to justify them using comments.
Section A : The abstract class MedicalPractitioner (5 marks)

All medical practitioners (doctors) working for Best Medicals are identified by their surname (assume no two doctors have the same surname). Their mobile phone numbers are also kept in the system allowing them to be contacted in case of an emergency. Hence this class should provide instance variables to store surname and phone, with associated accessors. The consultation fee is computed differently for generalists (GPs) and specialists but both are based on consultation time (measured in minutes). Hence the method to compute fees must be made abstract and must take as its argument an int value (consultation time in minutes).

(i) Write an abstract class named MedicalPractitioner with instance variables for surname and phone. Provide appropriate accessors for these instance variables. (2 marks)

(ii) Provide a constructor, taking as arguments values for surname and phone. (2 marks)

(iii) Provide an abstract method named double fee(int mins) to compute and return consultation fee, a double value. (1 mark)

// Write your code here

abstract class MedicalPractitioner {

Section B: The subclass GeneralPractitioner (GP) (7 marks)

Currently, general practitioners (GPs) charge a fee of $20, when consultation time is 5 minutes or less, and $30, when consultation time is 6 minutes or more. Also each GP charges an additional surcharge which is a flat amount, independent of consultation time. This surcharge reflects the experience and qualifications of a GP, and hence varies from one GP to another.

(i) Create a new class named GeneralPractitioner extending the MedicalPractitioner class adding appropriate instance variables for short consultation rate, long consultation rate and surcharge. The variables for short and long consultation rates must be initialized to 20 and 30 respectively, but methods should be provided to change them at a later time, if necessary. (2 marks)

(ii) Write a constructor that takes as arguments values for surname, phone and surcharge setting the corresponding instance variables either directly or by calling the superclass constructor. (2 marks)

(iii) Implement the method double fee(int mins) to compute and return the charge for consulting a specific GP for mins minutes. This method should also take into account the applicable surcharge. (3 marks)

// Write your code here

class GeneralPractitioner …
{
    …
Specialist fee is computed based on consultation time in minutes, but with the restriction that the minimum charge is $50 and the maximum charge is $100 per consultation. For example, if the rate charged by a specialist is $5/min, then the fee should be computed as follows:

<table>
<thead>
<tr>
<th>Time</th>
<th>Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 minutes</td>
<td>$50 (minimum fee), as 5 x $5 = $25 is less than $50</td>
</tr>
<tr>
<td>12 minutes</td>
<td>$60 as 12 x $5 = $60</td>
</tr>
<tr>
<td>30 minutes</td>
<td>$100 (maximum fee), as 30 x $5 = $150 is greater than $100</td>
</tr>
</tbody>
</table>

This class should also store the additional charge applicable for the initial visit of a patient. This charge is fixed for a given specialist but differs from one specialist to another, as it deals with the cost of initial lab tests, x-rays and admin related costs, etc.

(i) Create a new class named **Specialist** extending the **MedicalPractitioner** class adding appropriate instance variables for storing rate-per-minute and initial charge. Provide an appropriate accessor for getting the initial charge for a particular specialist. (2 marks)

(ii) Write a constructor that takes as arguments values for **surname**, **phone**, initial charge and rate-per-minute, setting the corresponding instance variables either directly or by calling the superclass constructor. (2 marks)

(iii) Implement the method **double fee(int mins)** to compute and return the charge for consulting a specific GP for **mins** minutes.. (3 marks)

// Write your code here

class Specialist …
{
    …
Section D: A simple Application Producing Billing Information (16 marks)

In this part, you are required to create a simple billing system for Best Medicals by using the MedicalPractitioner and its subclass developed in sections A-C, following the steps outlined below:

(i) Create an array of 4 MedicalPractitioner references named mps (as there are currently two general practitioners and two specialists) (2 marks)

(ii) Construct two GeneralPractitioner and two Specialist objects specified below storing their references in the array created in (i) above. (2 marks)

List of General Practitioners

<table>
<thead>
<tr>
<th>Surname</th>
<th>Phone</th>
<th>Surcharge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leary</td>
<td>0412-123123</td>
<td>$12.0</td>
</tr>
<tr>
<td>Newton</td>
<td>0412-234121</td>
<td>$10.0</td>
</tr>
</tbody>
</table>

List of Specialists

<table>
<thead>
<tr>
<th>Surname</th>
<th>Phone</th>
<th>Initial Charge</th>
<th>Rate/min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lee</td>
<td>0412-123456</td>
<td>$100</td>
<td>$12</td>
</tr>
<tr>
<td>Ratnam</td>
<td>0431-987456</td>
<td>$80</td>
<td>$5</td>
</tr>
</tbody>
</table>

(iii) This section requires you to produce the billing information for all the patients visiting the clinic each day. The billing information, which includes patient details and medical fees must be written to a text file named billing.txt as shown on the next page.

The program should run continuously, first reading patient details including name and medicare card number. It should terminate only when an empty patient name is entered. For each patient, the details of consultation must be read next, which include the surname of the medical practitioner and the duration of consultation. If the surname of the medical practitioner entered is not found in the lists above, users must be prompted to re-enter. If the medical practitioner is a specialist the system must query whether it is the first visit, and if so, the initial charge must be added to the billing information. The sample input/output and expected billing file is shown on the next page.

(12 marks)
Sample Input/Output

Enter patient name : C Tom
Enter medicare number : 3299-967344
Enter name of Doctor : Lee
Enter duration of consultation : 12
First visit Y/N ? N
Billing details written to file.

Enter patient name : H Beng
Enter medicare number : 3288-876546
Enter name of Doctor : Leary
No such name. Enter name of Doctor : Leary
Enter duration of consultation : 10
Billing details written to file.

Enter patient name : T Rahim
Enter medicare number : 3212-876544
Enter name of Doctor : Newton
Enter duration of consultation : 5
Billing details written to file.

Enter patient name : R Cooper
Enter medicare number : 3245-765432
Enter name of Doctor : Lee
Enter duration of consultation : 20
First visit Y/N ? Y
Billing details written to file.

Enter patient name : Empty line to terminate

Text written to billing.txt by sample input/output above.

<table>
<thead>
<tr>
<th>Patient</th>
<th>Med. No.</th>
<th>Doctor</th>
<th>Fee</th>
<th>Initial Charge (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C Tom</td>
<td>3299-967344</td>
<td>Lee</td>
<td>$100.0</td>
<td></td>
</tr>
<tr>
<td>H Bill</td>
<td>3288-876546</td>
<td>Leary</td>
<td>$42.0</td>
<td></td>
</tr>
<tr>
<td>T Rahim</td>
<td>3212-876544</td>
<td>Newton</td>
<td>$25.0</td>
<td></td>
</tr>
<tr>
<td>R Cooper</td>
<td>3245-765432</td>
<td>Lee</td>
<td>$100.0</td>
<td>$100.0</td>
</tr>
</tbody>
</table>
// Write your code here

import java.io.*;
import java.util.*;

public class Billing
{

Solutions

Part I  Multiple Choice

1.  C
2.  D
3.  C
4.  D
5.  A
6.  C
7.  B
8.  A
9.  B
10. C
11. C
12. B
13. A
14. D
15. B
16. D
17. B
18. B
19. C
20. C
Part II  Program Segments
In this section we are attempting to assess the students ability to use the basic control structures, arrays, classes, inheritance, polymorphism and exceptions.

21. Testing students understanding of loops
for (int i=1; i<=10; i++)
    System.out.print(" " + 1000 + i* 2 );

22.
int diff = carSpeed – zoneSpeed;
int fine = 0;
if (diff <= 10)
    fine = 120;
else if (diff <= 30)
    fine = 250;
else
    fine = 500;
System.out.println("Fine is “ + fine);

23. Objective: Testing students understanding of algorithms.
int numHyphens = 5;
int numStars = 1;
for (int i=1; i<=5; i++)
    for (int j=1; j<= numHyphens; j++)
        System.out.pint(" ");
    for (int j=1; j<= numStars; j++)
        System.out.pint("*");
    for (int j=1; j<= numHyphens; j++)
        System.out.pint(" ");
    System.out.println();
    numStars+= 2;
    numHyphens--;

24. Objective: Testing students understanding of array manipulation
highest = 0;
for (int i=0; i<m.length; i++)
    if (m[i] > highest)
        highest = m[i];

25. Objective: Testing students ability to read a program
26.
Objective: Testing students understanding of arrays and abstract classes and method overriding

class CSStudent extends Student
{
    public CSStudent(String name, String number)
    {
        super(name,number);
    }
    public void enrol(String course) throws Exception
    {
        if ( course.compareTo("Java01") != 0 )
            throw new Exception("Enrol in Java01 before " +course);
        super.enrol(course);
    }
    public void withdraw(String course) throws Exception
    {
        super.withdraw(course);
    }
}

class ITStudent extends Student
{
    public ITStudent(String name, String number)
    {
        super(name,number);
    }
    public void enrol(String course) throws Exception
    {
        if ( course.compareTo("Java01") == 0 )
            throw new Exception("InvalidCombination Java01 + " + course);
        super.enrol(course);
    }
}
Part III  Program Writing

Part A  Writing a class

Summary

Here students are required to write a simple abstract class. Students are expected to abide by general guidelines such as encapsulation by making all the instance variables private. Students are tested for normal methods, abstract methods and static methods. Avoid overlooking simple syntax errors such as missing semicolons.

Instance variables and accessors
Constructor
Abstract method for getFee

abstract class MedicalPractioner
{
    private String name;
    private String phone;

    public MedicalPractioner(String n, String p)
    {
        name = n;
        phone = p;
    }
    public String getName()
    {
        return name;
    }
    public String getPhone()
    {
        return phone;
    }
    public abstract double getFee(int mins);
}
**Part B Writing the subclass GP**

Here we are attempting to assess student’s ability to extend classes. Students should demonstrate the ability to call superclass methods or constructors. Avoid overlooking simple syntax errors such as missing semicolons.

Extending class, adding 1 instance var, 2 static vars adding & initializing Constructor calling superclass constructor and setting instance vars Implementing the abstract method grossSalary

class GeneralPractioner extends MedicalPractioner
{
    private static double rate1 = 20.0;
    private static double rate2 = 30.0;
    private double premium;

    public GeneralPractioner(String n, String p, double premium)
    {
        super(n,p);
        this.premium = premium;
    }

    public double getFee(int mins)
    {
        if (mins < 10)
            return rate1 + premium;
        else return rate2 + premium;
    }
}
**Part C Writing a subclass (Specialist)**

Here we are attempting to assess student’s ability to extend classes. Students should demonstrate the ability to call superclass methods or constructors. Student should also write a constructor and an accessor.

Avoid double penalty and overlook simple syntax errors such as missing semicolons.

Extending class, adding instance and accessor for initial
Constructor calling superclass constructor and setting instance vars
Implementing the abstract method grossSalary

class Specialist extends MedicalPractioner {
    private double initial;
    private double rate;

    public Specialist(String n, String p, double initial, double rate)
    {
        super(n,p);
        this.rate = rate;
        this.initial = initial;
    }
    public double getInitial()
    {
        return initial;
    }
    public double getFee(int mins)
    {
        double val = rate * mins;
        if ( val < 50.0 )
            val = 50.0;
        if (val > 100.0)
            val = 100.0;
        return val;
    }
}
Part D Simple application

Here we are attempting to assess student's ability to solve problems using an object-oriented paradigm. Students are required to construct objects storing them in an array, invoke methods on specified objects, write to files, use all the control structures, handle any exceptions, file handling etc. Avoid overlooking simple syntax errors such as missing semicolons.

class Billing
{
    private MedicalPractioner[] mps;
    int count;
    public Billing()
    {
        // (i) Create an array of 6 Employee references named emps
        mps = new MedicalPractioner[4];

        // (ii) Constructing Subclass objects & storing their references
        mps[0] = new GeneralPractioner("Leary","0412-123123",12.00);
        mps[1] = new GeneralPractioner("Newton","0412-234121",5.0);
        mps[2] = new Specialist("Lee","0412-123456",100.00,5.00);
        mps[3] = new Specialist("Ratnam","0431-987456",80.00,4.50);
    }

    // loop until empty patient name
    // reading patient details
    // reading doctor name and consultation time
    // validating doctor name
    // getting additional info for specialist
    // handling initial visit for specialist
    // writing to file

    public void compute() throws IOException
    {
        PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter("billing.txt")));
        Scanner sc = new Scanner(System.in);
        pw.println("Patient Med. No. Doctor Fee Initial Charge");
        pw.println("________ _____________________________");
        do {
            System.out.print("Enter patient name : ");
            String pName = sc.nextLine();
            if (pName.length()==0)
                break;
            System.out.print("Enter medicare number : ");
            String mcNum = sc.nextLine();
            do {
                System.out.print("Enter name of Doctor : ");
                String dName = sc.nextLine();
                int i;
                for (i=0; i<mps.length; i++)
                {
                    if (mps[i].getName().compareTo(dName)==0)
                    {
                        break;
                    }
                }
                System.out.print("Enter reason of visit : ");
                String reason = sc.nextLine();
                System.out.print("Enter additional info: ");
                String info = sc.nextLine();
                System.out.print("Enter additional chars: ");
                String chars = sc.nextLine();
                System.out.print("Enter initial charge: ");
                double charge = sc.nextDouble();
                pw.println(pName + " " + mcNum + " " + dName + " " + reason + " " + info + " " + chars + " " + charge);
            }
        }
    }
}
{ System.out.print("Enter duration of consult.: ");
    int mins = sc.nextInt();
    sc.nextLine();
    pw.println(pName+"\t"+mcNum+"\t"+mps[i].getName()+"\t"+mps[i].getFee(mins));
    if (mps[i] instanceof Specialist )
    {
        System.out.print("Fisrt visit Y/N ?");
        String resp = sc.nextLine();
        if ( resp.compareTo("Y") == 0)
            pw.println("\t"+((Specialist)mps[i]).getInitial());
    }
    pw.println();
    break;
}
if ( i == mps.length )
    System.out.print("No such name. Reneter.");
else break;
} while (true);
while (true);
pw.close();

public static void main(String args[]){throws IOException
{
    Billing b = new Billing();
    b.compute();
}