**T323  
ICT104 Program Design and Development**

**Week 5 Tutorial**

**Topic: Object Oriented Concepts**

**Objective: Students learn about the basics of Objects and Classes using Java**

**Submission: The completed tutorial word file containing student’s answers need to be  
uploaded to Moodle by Sunday 3/12/2023 @ 23.59.**

**Part A:** Highlight the correct option.

1. When declaring class data members, it is best to declare them as:

A) private members

B) public members

C) protected members

D) restricted members

2. When a subclass overrrides a superclass method:

A) Both methods may be called with a subclass object

B) Only the subclass method may be called with a subclass object

C) Only the superclass method may be called with a subclass object

D) Neither method may be called with a subclass object

3. If two methods have the same name but different signatures, they are:

A) overridden

B) overloaded

C) superclass methods

D) subclass methods

4. If ClassA extends ClassB, then:

A) public and private members of ClassB are public and private, respectively, in ClassA

B) public members in ClassB are public in ClassA, but private members in ClassB cannot be directly accessed in ClassA

C) neither public or private members in ClassB can be directly accessed in ClassA

D) private members in ClassB are changed to protected members in ClassA

5. Which of the following statements declares Salaried as a subclass of PayType?

A) public class Salaried extends PayType

B) public class Salaried implements PayType

C) public class Salaried derivedFrom(Paytype)

D) public class PayType derives Salaried

6. In an inheritance relationship:

A) The superclass constructor always executes before the subclass constructor

B) The subclass constructor always executes before the superclass constructor

C) The constructor with the lowest overhead always executes first regardless of inheritance

D) The unified constructor always executes first regardless of inheritance

7. What key word can you use to call a superclass constructor explicitly?

A) goto

B) this

C) super

D) extends

8. The super statement that calls the superclass constructor:

A) must be the first statement in the superclass's constructor

B) can appear in any method of the subclass

C) must be the first statement in the subclass's constructor

D) is deprecated and is no longer supported in newer versions of Java

9. Replacing inadequate superclass methods with more suitable subclass methods is known as what?

A) Method upgrading

B) Tactical inheritance

C) Method overriding

D) Method overloading

10. Look at the following code.

Line 1 public class ClassA

Line 2 {

Line 3 public ClassA() {}

Line 4 public void method1(){}

Line 5 }

Line 6 public class ClassB extends ClassA

Line 7 {

Line 8 public ClassB(){}

Line 9 public void method1(){}

Line 10 }

Line 11 public class ClassC extends ClassB

Line 12 {

Line 13 public ClassC(){}

Line 14 public void method1(){}

Line 15 }

Which method1 will be executed as a result of the following statements?

ClassA item1 = new ClassC();

item1.method1();

A) Line 4

B) Line 9

C) Line 14

D) This is an error and will cause the program to crash.

11. In the following statement, which is the superclass?

public class ClassA extends ClassB implements ClassC

A) ClassA

B) ClassB

C) ClassC

D) Cannot tell

12. Look at the following code. The method in line \_\_\_\_\_\_\_\_ will override the method in line \_\_\_\_\_\_\_\_.

Line 1 public class ClassA

Line 2 {

Line 3 public ClassA() {}

Line 4 public int method1(int a){}

Line 5 public int method2(int b){}

Line 6 }

Line 7 public ClassB extends ClassA

Line 8 {

Line 9 public ClassB(){}

Line 10 public int method1(int b){}

Line 11 public int method2(double c){}

Line 12 }

A) 4, 10

B) 5, 11

C) 10, 4

D) 11, 5

**Part B:** State TRUE or FALSE:

1. If two methods in the same class have the same name but different signatures, the second overrides the first.
2. It is not possible for a superclass to call a subclass's method.
3. Inheritance involves a subclass, which is the general class, and a superclass, which is the specialized class.
4. In an inheritance relationship, the subclass constructor always executes before the superclass constructor.

**Part C:** Write Java Code for the following case scenarios:

1. Design a class named Employee. The class should keep the following information in fields:

* Employee name
* Employee number in the format XXX-L, where each X is a digit within the range 0-9 and the L is a letter within the range A\_M.
* Hire date

Write one or more constructors and the appropriate accessor and mutator methods for the class.

Next, write a class named ProductionWorker that extends the Employee class. The ProductionWorker class should have fields to hold the following information:

* Shift (an integer)
* Hourly pay rate (double)

The workday is divided into two shifts: day and night. The shift field will be an integer value representing the shift that the employee works. The day shift is shift 1 and the night shift is shift 2. Write one or more constructors and the appropriate accessor and mutator methods for the class. Demonstarte the classes by writing a program that uses a ProductionWorker object.

1. In a particular factory, a team leader is an hourly paid production worker that leads a small team. In addition to hourly pay, team leaders earn a fixed monthly bonus. Team leaders are required to attend a minimum number of hours of training per year. Design a TeamLeader class that extends the ProductionWorker class you designed in the previous question. The TeamLeader class should have fields for the monthly bonus amount, the required number of training hours, and the number of training hours that the team leader has attended. Write one or more constructors and the appropriate accesors and mutator methods for the class. Demonstrate the class by writing a program that uses a TeamLeader object

3. Identify the output of the following Java code. Justify the syntax.

File Name: CompSciStudentDemo.java

/\*\*

This program demonstrates the CompSciStudent class.

\*/

public class CompSciStudentDemo

{

public static void main(String[] args)

{

// Create a CompSciStudent object.

CompSciStudent csStudent =

new CompSciStudent("Jennifer Haynes",

"167W98337", 2015);

// Store values for math, CS, and gen ed hours.

csStudent.setMathHours(12);

csStudent.setCsHours(20);

csStudent.setGenEdHours(40);

// Display the student's data.

System.out.println(csStudent);

// Display the number of remaining hours.

System.out.println("Hours remaining: " +

csStudent.getRemainingHours());

}

}

File Name: CompSciStudent.java

/\*\*

This class holds data for a computer science student.

\*/

public class CompSciStudent extends Student

{

// Required hours

private final int MATH\_HOURS = 20; // Math hours

private final int CS\_HOURS = 40; // Comp sci hours

private final int GEN\_ED\_HOURS = 60; // Gen ed hours

// Hours taken

private int mathHours; // Math hours taken

private int csHours; // Comp sci hours taken

private int genEdHours; // General ed hours taken

/\*\*

The Constructor sets the student's name,

ID number, and the year admitted.

@param n The student's name.

@param id The student's ID number.

@param year The year the student was admitted.

\*/

public CompSciStudent(String n, String id, int year)

{

super(n, id, year);

}

/\*\*

The setMathHours method sets the number of

math hours taken.

@param math The math hours taken.

\*/

public void setMathHours(int math)

{

mathHours = math;

}

/\*\*

The setCsHours method sets the number of

computer science hours taken.

@param cs The computer science hours taken.

\*/

public void setCsHours(int cs)

{

csHours = cs;

}

/\*\*

The setGenEdHours method sets the number of

general ed hours taken.

@param genEd The general ed hours taken.

\*/

public void setGenEdHours(int genEd)

{

genEdHours = genEd;

}

/\*\*

The getRemainingHours method returns the

the number of hours remaining to be taken.

@return The hours remaining for the student.

\*/

@Override

public int getRemainingHours()

{

int reqHours, // Total required hours

remainingHours; // Remaining hours

// Calculate the required hours.

reqHours = MATH\_HOURS + CS\_HOURS + GEN\_ED\_HOURS;

// Calculate the remaining hours.

remainingHours = reqHours - (mathHours + csHours

+ genEdHours);

return remainingHours;

}

/\*\*

The toString method returns a string containing

the student's data.

@return A reference to a String.

\*/

@Override

public String toString()

{

String str;

str = super.toString() +

"\nMajor: Computer Science" +

"\nMath Hours Taken: " + mathHours +

"\nComputer Science Hours Taken: " + csHours +

"\nGeneral Ed Hours Taken: " + genEdHours;

return str;

}

}

File Name: Student.java

/\*\*

The Student class is an abstract class that holds

general data about a student. Classes representing

specific types of students should inherit from

this class.

\*/

public abstract class Student

{

private String name; // Student name

private String idNumber; // Student ID

private int yearAdmitted; // Year admitted

/\*\*

The Constructor sets the student's name,

ID number, and year admitted.

@param n The student's name.

@param id The student's ID number.

@param year The year the student was admitted.

\*/

public Student(String n, String id, int year)

{

name = n;

idNumber = id;

yearAdmitted = year;

}

/\*\*

The toString method returns a String containing

the student's data.

@return A reference to a String.

\*/

public String toString()

{

String str;

str = "Name: " + name

+ "\nID Number: " + idNumber

+ "\nYear Admitted: " + yearAdmitted;

return str;

}

/\*\*

The getRemainingHours method is abstract.

It must be overridden in a subclass.

@return The hours remaining for the student.

\*/

public abstract int getRemainingHours();

}

**Optional Task**

Write a Java code for a class named **SportsVenuePass**. This class will be used to instantiate objects that represent the right to access a performance in the sports venue. The class will have **two (2) private instance variables**: a **boolean** variable named ***used*** and an **integer** variable named ***passId***. The boolean variable *used* will be initialised to false and changed to true by the method **useThisPass()** which will be called when the pass is presented at the sports ground. When the pass is created, a unique new id number is allocated to ***passId***. This id number will be generated by adding one to the previously used ***passId*** number, which is kept stored in a variable shared by all **SportsVenuePass** objects.