5COSC019W - Tutorial 5 Exercises

1 Static Methods

Implement a utility class `CalcManager` which provides the following `static` methods:

- `isEven(int n)`: returns `true` if `n` is even, otherwise it returns false.
- `cube(int n)`: returns the cube of `n`.
- `add(double...x)`: returns the sum of all its given arguments. The number of `double` arguments passed to it is arbitrary (i.e. any number of arguments can be passed to it).

2 Static Fields

1. Guess what is the output of the following program `EmployeeManager`, without running it. The program manages a number of employees.

```
class Employee {
    static int numberOfEmployees;
    static Address company_address;
    String name;

    Employee (String name1, Address co_address) {
        name = name1;
        company_address = co_address;

        numberOfEmployees++;
    }

    public void print() {
        System.out.print("Name: " + name + ", ");
        company_address.print();
    }
}


class Address {
    String street;
    int number;
    String city;

    Address(String street1, int number1, String city1) {
        street = street1;
        number = number1;
```

```
        city = city1;
    }

    void print() {
        System.out.println(number + " " + street + " " + city);
    }
}

public class EmployeeManager {
    public static void main(String[] args) {
        Address a1 = new Address("Watford Road", 101, "London");
        Address a2 = new Address("Guilford Road", 5, "York");
        Address a3 = new Address("Exhibition Road", 77, "London");

        a1 = a2;
        Employee e1 = new Employee("John", a1);
        Employee e2 = new Employee("George", a2);
        Employee e3 = new Employee("Helen", a3);

        System.out.print("e1 contains: ");
        e1.print();
        System.out.print("e2 contains: ");
        e2.print();
        System.out.print("e3 contains: ");
        e3.print();

        e1.company_address = a2;

        System.out.print("\nAfter e1.company_address = a2\n");

        System.out.print("e1 contains: ");
        e1.print();
        System.out.print("e2 contains: ");
        e2.print();
        System.out.print("e3 contains: ");
        e3.print();

        System.out.println("\nMemory location for e1.company_address object: " +
                           e1.company_address);
        System.out.println("Memory location for e2.company_address object: " +
                           e2.company_address);
        System.out.println("Memory location for e3.company_address object: " +
                           e3.company_address);

        System.out.println("\ne1.numberOfEmployees: " + e1.numberOfEmployees);
        System.out.println("e2.numberOfEmployees: " + e2.numberOfEmployees);
        System.out.println("e3.numberOfEmployees: " + e3.numberOfEmployees);
    }
}
```

2. Verify your guess by running the program. Make sure you understand why the specific output is displayed, if you did not get it right.

3. What is the limitation of the given program, considering that field `company_address` is `static` in `Employee`? Can the above code create employees working for different companies?

3 Using super() - Implementing the toString() method

1. Fill in the missing line indicated by the comment "Missing line goes here" in the following code.

   The missing statement should call the constructor of the parent class.

   Hint: The `super` keyword should be used to call a method on the parent class.

```
class Book {
    private int pages;  // number of pages in the book

    Book(int pages) {
        this.pages = pages;
    }

    public String toString() {
        return ""+pages;  // convert pages to String before returning
    }
}

class Dictionary extends Book {
    private int words;  // number of words in the dictionary

    Dictionary(int words, int pages) {
        // missing line 1 goes here - should call the parent constructor here
        this.words = words;
    }

    /*** for subquestion 2 implement toString() here ***/
}

public class BookTest {
    public static void main(String[] args) {
        Dictionary d1 = new Dictionary(100000, 500);
        System.out.println(d1);
    }
}
```

2. Implement method `toString()` inside class `Dictionary`. The method should return as a String the concatenation of the value of field `words` and the value of field `pages` inherited from the parent class `Book`.

   Hint: Since `pages` is `private` in `Book`, it cannot be accessed directly by `Dictionary`. What methods of `Book` can be called to print the value of that field?

4 Final Classes and Methods

What is wrong with the following program? Make the necessary corrections to compile the code.

```
public final class X1 {

}

class X2 extends X1 {
    public final void foo() {
        System.out.println("foo() called in X2");
    }
}

class X3 extends X2 {
    // overriding foo() of X2
    public void foo() {
        System.out.println("foo() called in X3");
    }
}
```

5 Construction of Objects

1. Guess what will be the output of the following program without running it. Verify your guess by running the program, and make sure you understand why the specific output is displayed.

```
class Cell {
    Cell() {
        System.out.println("Cell constructor called");
    }
}

class TinyCell extends Cell {
    TinyCell() {
        System.out.println("TinyCell constructor called");
    }
}

class MicroscopicCell extends TinyCell {
    MicroscopicCell() {
        System.out.println("MicroscopicCell constructor called");
    }
}


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

```
        Cell c = new MicroscopicCell();
    }
}
```

2. What is wrong with the following code? What will happen and why, if you add the line "`super(5);`" (without the quotes!) at the point indicated by the comment "missing line goes here"?

```
class Cell2 {
    int x;

    Cell2(int x) {
        this.x = x;
        System.out.println("Cell2 constructor called");
    }
}

class TinyCell2 extends Cell2 {
    TinyCell2() {
        // missing line goes here
        System.out.println("TinyCell2 constructor called");
    }
}

class MicroscopicCell extends TinyCell2 {
    MicroscopicCell() {
        System.out.println("MicroscopicCell constructor called");
    }
}


public class CellTest2 {
    public static void main(String[] args) {
        Cell2 c = new MicroscopicCell();
    }
}
```

6 Using the instanceof operator

Guess what will be the output of the following program without running it. Then verify your guess by running the program.

```
interface Node {
    void print();
}

class TreeNode implements Node {
    int data;
```

```
    TreeNode(int data) {
        this.data = data;
    }

    public void print() {
        System.out.println("data: " + data);
    }
}

class Vertex implements Node {
    int x;  // x coordinate
    int y;  // y coordinate

    Vertex(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void print() {
        System.out.println("x: " + x + ", y: " + y);
    }
}

public class NodeTest {
    static void printLabel(Node n) {
        if (n instanceof TreeNode)
            System.out.print("Treenode-> ");
        else if (n instanceof Vertex)
            System.out.print("Vertex-> ");

        n.print();
    }

    public static void main(String[] args) {
        Node n1 = new TreeNode(100);
        Node n2 = new Vertex(500, 400);
        printLabel(n1);
        printLabel(n2);
    }
}
```

7 Challenge: A Clock Program: Implementing Class Hierarchies - Inheritance

Implement a class `Clock` whose getHours and getMinutes methods return the current time at your location. (use `java.time.Instant.now().toString()`)

Also provide a `getTime()` method that returns a string with the hours and minutes by calling the `getHours` and `getMinutes` methods. Provide a subclass `WorldClock` whose constructor accepts a time offset. For example, if you live in California, a `new WorldClock(3)` should show the time in New York, three time zones ahead. Which methods did you over- ride? (You should not override `getTime`.)

Add an alarm feature to the Clock class. When `setAlarm(hours, minutes)` is called, the clock stores the alarm in a field. When you call `getTime`, and the alarm time has been reached or exceeded, return the time followed by the string "Alarm" and clear the alarm. What do you need to do to make the `setAlarm` method work for `WorldClock` objects?