Faculty of Science and Engineering

COMP125 Fundamentals of Computer Science
Workshop Week 3

Learning outcomes

By the end of this session, you will know some of Java basics. In particular, you will be able to design and write simple Java classes.

Questions

1. **Import-Export**

   It is important to know how to import Java projects from archive files (.jar/ .zip) and how to export your project(s) into archive files. First. we'll learn how to import a project.

   a. Click "File" –> "Import" –> "Existing Projects into Workspace"
   b. Select option "Select Archive file" and click on "Browse"
   c. Choose the archive files (".zip") that contains project(s) you want to open. Please note an archive file may contain multiple projects and click "ok"
   d. Check all projects you want to import and click "Finish"

   Practice the above procedure using file `classesAndObjectsProgram.zip` file uploaded on iLearn under Section 3. You should see a project `classesObjectsProject` if correctly imported.

   Next, we'll learn how to export a project.

   a. Click "File" –> "Export" –> "General" –> "Archive file"
   b. Select all projects you want to export in the archive file in the left panel
   c. In the "To archive file" section, choose file path and name and click "Finish"

   Export the project `classesObjectsProject` to an archive file `<studentID>.zip` where <studentID> is your Student ID.

2. (Problem solving and loops) Write a function (or method) that when passed an integer, returns the number of times it has to be divided by two, to reach zero. For example, 19 (or -19) needs to be divided by two, five times, to reach zero.

   a. 19/2 = 9,
   b. 9/2 = 4,
   c. 4/2 = 2,
   d. 2/2 = 1,
   e. 1/2 = 0

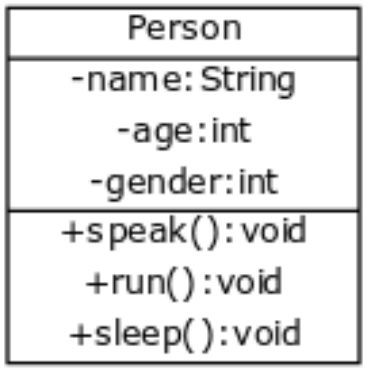
Solution:

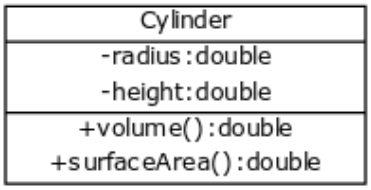
```
public static int countDivisonsByTwo(int an) {
        int count = 0;
        while(n != 0) {
                count++;
                n/=2;
        }
        return count;
}
```

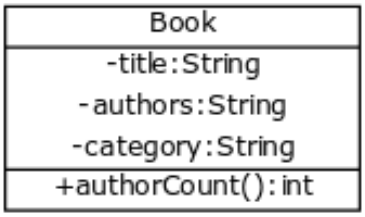
3. Design classes (no implementation) that encapsulate the following real life entities. Add up to three data members for each class. Select the three most important attributes if you think a class has more than three attributes. Describe your design in terms of a UML class diagram as shown in week 2 lecture.

   a. Person
   b. Cylinder
   c. Book

Solution:







4. (a) Consider the following class definition,

```
1 public class Car {
2         public String model;
3         public int price;
4 }
```

Declare and instantiate an object `myCar` of class `Car`. Assign the value `Corolla` to the data member `model` and the value 21999 to the data member `price` of object `myCar`.

Solution:

```
1 Car myCar = new Car();
2 myCar.model = ``Corolla'';
3 myCar.price = 21999;
```

(b) Consider the following class definition,

```
1 public class Date {
2         public int day, month, year;
3 }
```

Declare and instantiate an object `graduation` of class `Date`. Assign values to data members of object `graduation` such that it represents the date 13th April, 2011.

Solution:

```
Date graduation = new Date();
graduation.day = 13;
graduation.month = 4;
graduation.year = 2011;
```

5. (a) Consider the following class definition,

```
public class Time {
        public int hour, minute, second;
}
```

Explain why it's a bad idea for the data members to be public, by writing a client that is malicious and assigns invalid values to the daa members of `Time` object.

Solution:

```
Time myTime = new Time();
time.hour = 888;
time.minute = -54;
time.second = -1729;
```

(b) Solve the problem of `public` data members in the previous part by first changing visibility of the data members of class `Time` to `private` and then adding getters and setters. The setter for `hour` should constrain the passed value in the range [0, 23]. That is, if the passed value is less than 0, `hour` should be 0, otherwise if the passed value is more than 23, `hour` should be 23, otherwise `hour` should become the passed value. Similarly, the setters for `minute` and `second` should constrain the passed value in the range [0, 59].

Solution:

```
public class Time {
        private int hour, minute, second;

        //setters
        public void setHour(int hour)  {
                if(hour >= 0 && hour <=23)
                        this.hour = hour;
                else
                        this.hour = 0;
        }

        public void setMinute(int minute)  {
                if(minute >= 0 && minute <=59)
                        this.minute = minute;
                else
                        this.minute = 0;
        }

        public void setSecond(int second) {
                if(second >= 0 && second <=59)
                        this.second = second;
                else
                        this.second = 0;
        }

        //getters
        public int getHour() {
                return hour;
        }
```

```
30
31          public int getMinute() {
32                  return minute;
33          }
34
35          public int getSecond() {
36                  return second;
37          }
38  }
```

(c) Declare, instantiate an object `t1` of class `Time` written in the previous part. Assign values to the data members such that it represents the time 19:30:45 (half past seven in the evening and another 45 seconds).

Solution:

```
1  Time t1 = new Time();
2  t1.setHour(19);
3  t1.setMinute(30);
4  t1.setSecond(45);
```

(d) Declare and instantiate an object `t2` of class `Time` written in the previous part. Assign 95 to `hour`, -78 to minute, and 55 to second. Display all data members on the console. What time would `t2` represent?

Solution:

```
1  Time t2 = new Time();
2  t2.setHour(95);
3  t2.setMinute(-78);
4  yourTime.setSecond(55);
5  System.out.println(t2.getHour()); //23
6  System.out.println(t2.getMinute()); //0
7  System.out.println(t2.getSecond()); //55
```

(e) List the mistakes (syntactical and logical) in the following constructor for class `Time` -

```
1  public void time(int h) {
2          hour = h;
3          minute = 0;
4          second = 0;
5  }
```

Solution:

- Constructor has no return type
- Name of constructor should be **exactly** the same as the class name.
- Constructor should use setters to assign values to data members.

Fixed constructor,

```
1  public Time(int h) {
2          setHour(0);
3          setMinute(0);
4          setSecond(0);
5  }
```

(f) Add two constructors to class `Time` with the following requirements:

- A constructor that is passed three parameters, one for each data member.

- A constructor that is passed two parameters, for hour and minute, and sets seconds to 0.

Solution:

```
public Time(int h, int m, int s) {
        setHour(h);
        setMinute(m);
        setSecond(s);
}

public Time(int h, int m, int s) {
        setHour(h);
        setMinute(m);
        setSecond(0); //default value
}
```

(g) Assuming the two constructors have been added to class `Time` according to previous part. Will the following program run successfully, or result in a compilation error? Explain your answer. Also, if there is a compilation error, what should be done to fix it?

```
Time ourTime = new Time();
```

Solution: It will result in a compilation error, since once parameterized constructors are defined, Java expects us to define the default constructor as well, and the default constructor that Java provides is no longer valid. The solution, therefore, is to add a default constructor.

```
public Time() {
        setHour(0);
        setMinute(0);
        setSecond(0);
}
```

6. **Assessed exercise**

a. Write a class definition for a `Line` in the first quadrant, as represented by its end points (x_1, y_1) and (x_2, y_2) such that $x_1, y_1, x_2, y_2 \geq 0$. It should include,

(a) Correct class header.
(b) Data members with appropriate visibility and data types.
(c) Getters
(d) Setters
(e) Constructors
  i. With no parameters. First and second points should be $(0,0)$.
  ii. With two parameters for (x_1, y_1). Second point should be $(0,0)$.
  iii. With four parameters for $(x_1, y_1), (x_2, y_2)$.
(f) A method `getLength()` that returns the length of the line as computed by the pythagorean formula

$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.
You can compute square root of a numerical value *val* by using `Math.sqrt(val)`.
(g) The method `toString()` that returns the `String` representation of object of class `Line`. For example, if a line goes from (1.5, 2.5) to (8, 6.5), the method should return "`(1.5, 2.5) to (8, 6.5)`".

b. Write a piece of client code that -

(a) Declare and instantiate an object `myLine` of class `Line` that represents a line from (4.2, 3.5) to ((6.5, 1.2)
(b) Display the details of the object `myLine`. Which method is used? Do you call that method explicitly?
(c) Store the length of `myLine` in variable `myLength` and display it on the console.

(d) Change one of the end-points of the line from the previous coordinate of (4.2, 3.5) to a new coordinate (1.5, 7.5), and display the details of the line once more.

Solution:

```
public class Line {
        private double x1, y1, x2, y2;

        public double getX1() {
                return x1;
        }

        public void setX1(double x1) {
                this.x1 = Math.abs(x1);
        }

        public double getY1() {
                return y1;
        }

        public void setY1(double y1) {
                this.y1 = Math.abs(y1);
        }

        public double getX2() {
                return x2;
        }

        public void setX2(double x2) {
                this.x2 = Math.abs(x2);
        }

        public double getY2() {
                return y2;
        }

        public void setY2(double y2) {
                this.y2 = Math.abs(y2);
        }

        public Line() {
                setX1(0);
                setY1(0);
                setX2(0);
                setY2(0);
        }

        public Line(double x1, double y1) {
                setX1(x1);
                setY1(y1);
                setX2(0);
                setY2(0);
        }

        public Line(double x1, double y1, double x2, double y2) {
                setX1(x1);
                setY1(y1);
                setX2(x2);
                setY2(y2);
        }

        public double getLength() {
                return Math.sqrt(Math.pow(x2-x1, 2) + Math.pow(y2-y1, 2));
        }

        public String toString() {
                return ``(''+x1+``,''+y1+``) to (''+x2+``, ''+y2+``)'';
        }
}
```

Solution:

```
public class Client {
        public static void main(String[] args) {
                Line myLine = new Line(4.2, 3.5, 6.5, 1.2);
                System.out.println(myLine); //displays details using toString()
                double myLength = myLine.getLength();
                System.out.println(myLength);
                myLine.setX1(1.5);
                myLine.setY1(7.5);
                System.out.println(myLine); //displays details using toString()
        }
}
```