MACQUARIE University

Department of Computing

COMP125 Fundamentals of Computer Science
Workshop - Workshop - Classes and Objects 1, JUnit

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).

   For this exercise, download the file `week3.zip` from iLearn but DO NOT unzip/open it.

   a. Click "File" –> "Import" –> "Existing Projects into Workspace" **and NOT "Archive file"**.
   b. Select option "Select Archive file" and click on "Browse"
   c. Choose the archive file ("week3.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
   e. Click "Finish"

   You should see a project `week3workshop` if correctly imported.

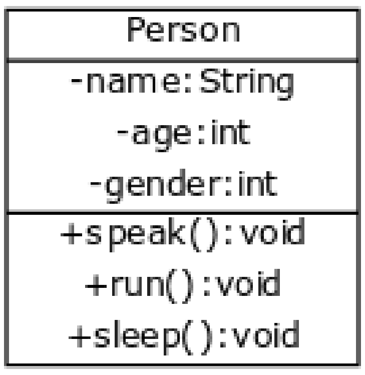
2. Design classes (no implementation) that encapsulate the following real life entities. Add up to three instance variables for each class. Select the three most important attributes if you think a class has more than three attributes. Describe your design in terms of a UML class diagram as shown in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

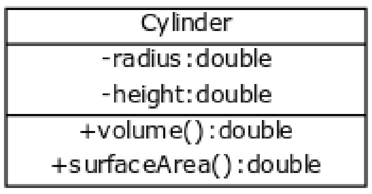
   a. `Person`

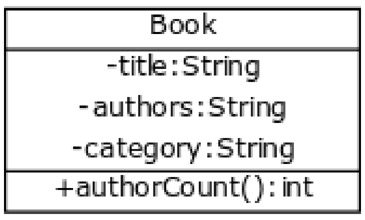
   b. `Cylinder`

   c. `Book` **(Take-home exercise)**

Solution:







3. (a) Consider the following class definition,

```java
public class Date {
        public int day, month, year;
}
```

In a client code (outside the class `Date`), create an object to represent 13th April, 2011.

Solution:

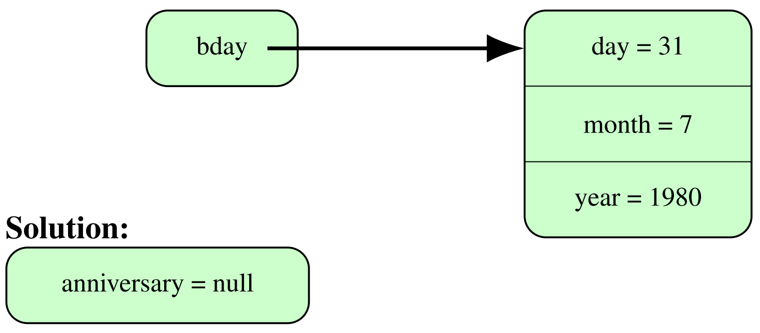
```java
Date graduation = new Date();
graduation.day = 13;
graduation.month = 4;
graduation.year = 2011;
```

(b) For the same class definition (`Date`), draw the memory diagram that represents the following objects. Please refer to the lecture notes to see the representation scheme we have established.

```java
Date bday = new Date();
bday.day = 31;
bday.month = 7;
bday.year = 1980;

Date anniversary = null;
```

Solution:



(c) **(Take-home exercise)** Consider the following class definition,

```java
public class Car {
        public String model;
        public int price;
}
```

In a client code (outside the class `Car`), create an object to represent a Toyota Corolla priced at $21,999.

Solution:

```java
Car myCar = new Car();
myCar.model = "Toyota␣Corolla";
myCar.price = 21999;
```

4. (a) Consider the following class definition,

```java
public class Person {
        public String name;
        public int age;
}
```

With the help of an example, explain why it's a bad idea for the instance variables to be public.

Solution:

```java
Person p = new Person();
p.name = "Hagrid";
p.age = -6; //OOPS!
```

(b) Solve the problem of `public` instance variables in the previous part by first changing visibility of the instance variables of class `Person` to `private` and then adding getters and setters. Age of a person cannot be negative. If the user tries to assign a negative value to a person's age, the person's age should be set to 0.

Solution:

```java
public class Person {
        private String name;
        private int age;

        //setters
        public void setName(String str) {
                name = str;
        }

        public void setAge(int a) {
                if(a <= 0)
                        age = 0;
                else
                        age = a;
        }

        public String getName() {
                return name;
        }

        public int getAge() {
                return age;
        }
}
```

(c) Declare and instantiate an object `baddy` of class `Person` written in the previous part, representing a person named "Tom Marvolo Riddle" aged 71.

Solution:

```java
Person baddy = new Person();
baddy.setName("Tom␣Marvolo␣Riddle");
baddy.setAge(71);
```

(d) Declare and instantiate an object `goody` of class `Person` written in the previous part, representing a person named "Luna Lovegood". Try and assign the value -6 to her age.

Display the values of the instance variables on the console.

Solution:

```java
Person goody = new Person();
goody.setName("Luna␣Lovegood");
goody.setAge(6);
System.out.println(goody.getName()); //"Luna Lovegood"
System.out.println(goody.getAge()); //0
```

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

```java
public void person(String s, int a) {
        name = s;
        age = a;
}
```

Solution:

- Constructor should have no return type, not even void.
- Name of constructor should be **exactly** the same as the class name. So, `Person`, not `person`.
- Constructor should use setters to assign values to instance variables.

Fixed constructor,

```java
public Person(String s, int a) {
        setName(s);
        setAge(a);
}
```

(f) Add a constructor to class `Person` with a single parameter for the instance variable name. The instance variable `age` should be set to 21.

Solution:

```java
public Person(String s) {
        setName(s);
        setAge(18); //setter is used here as a best practice.
}
```

(g) Assuming the constructor have been added to class `Person` 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?

```java
Person p = new Person();
```

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.

```java
public Person() {
        setName("anonymous");
        setAge(0);
}
```

5. **JUnit Testing**

We can test the correctness of individual methods through running JUnit tests on them.

In the project that you imported, open file `RectangleTest.java` and run it. You will see a green bar and a message that 5 out of 5 tests have passed (0 errors and 0 failures).

Now open the file `AllInOneTest.java` and run the tests. You will see that two tests pass while one fails. The tests that pass are:

a. `testIsSquare`: This tests the method `isSquare` from class `AllInOne`

b. `testAllSquares`: This tests the method `allSquares` from class `AllInOne`

The test that fails is `testCountPositiveEvens`: This tests the method `countPositiveEvens` from class `AllInOne`.

Your job is to complete the method `countPositiveEvens` so the test `testCountPositiveEvens` passes. **IMPORTANT:** note that the method should return 0 if the array passed is `null`. This is the first thing you should check in the method.

Solution: Refer to project in Eclipse archive file week3solution.zip

6. **compareTo method**

Consider the class `GoalScoringRecord` in the project imported. Complete the method `compareTo` so the corresponding test supplied in class `GoalScoringRecordTest` passes.

Solution: Refer to project in Eclipse archive file week3solution.zip