IIP
Test Unit 5 - Possible solution
Year 2016-2017

Name:

1. (3 points) Given the `LightBulb` class with the following structure:

```
public class LightBulb {
  private int lumen;
  private char energyClass;
  public static final char A='*', B='+', C='/', D='-';

  public LightBulb(int l, char e) {
    if (l>0 && (e==A || e==B || e==C || e==D)) { lumen=l; energyClass=e; }
  }
  ...
}
```

Implement the following methods:

a) (1.5 points) An overriden version of the `toString` method that returns a representation of the lightbulb in the format "`CLASS` light bulb of `LUMEN` lm", where `CLASS` depends on the value of `energyClass`:

A `LED`
B `Low consumption`
C `Halogen`
D `Incandescent`

and `LUMEN` is the `lumen` value. **You must employ the constant attributes when comparisons on energy class are done.**

b) (1.5 points) A `validLight` method that receives as parameter the maximum number of watts that a lamp supports (integer value) and returns if the lightbulb is suitable for that lamp (boolean value). The lightbulb is suitable if its consumption in watts is lower than the lamp max support. Consumption of a light bulb is calculated according to the following rules:

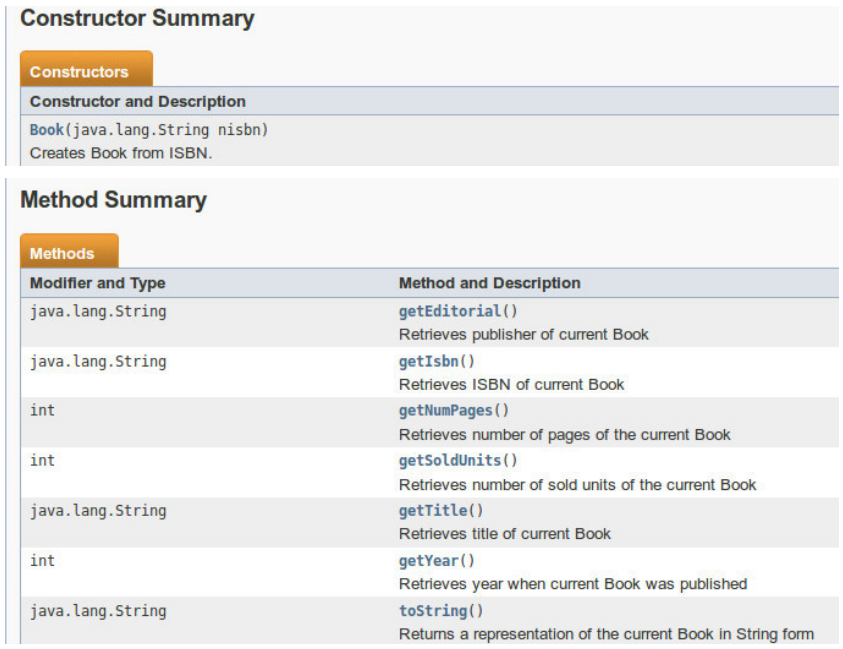
- Incandescent light bulbs have a consumption equal to number of lumens divided by 5
- Halogen light bulbs save a 20 % of consumption with respect to incandescent
- Low consumption light bulbs save a 65 % of consumption with respect to incandescent
- LED light bulbs save a 90 % of consumption with respecto to incandescent

```
public String toString() {
  String res;
  switch(energyClass) {
    case A: res+="LED"; break;
    case B: res+="Low consumption"; break;
    case C: res+="Halogen"; break;
    case D: res+="Incandescent"; break;
  }
  return res + " light bulb of " + lumen + "lm";
}

public boolean validLight(int maxWatts) {
  double baseCons = lumen/5.0;
  switch(energyClass) {
    case A: baseCons = baseCons * 0.1;
    case B: baseCons = baseCons * 0.35;
    case C: baseCons = baseCons * 0.8;
  }

  return (baseCons < maxWatts);
}
```

2. (7 points) Given a `Book` class with the following documentation:



Constructor Summary

Constructors

Constructor and Description
`Book(java.lang.String nisbn)`   Creates Book from ISBN.

Method Summary

Methods

Modifier and Type	Method and Description
`java.lang.String`	`getEditorial()` Retrieves publisher of current Book
`java.lang.String`	`getIsbn()` Retrieves ISBN of current Book
`int`	`getNumPages()` Retrieves number of pages of the current Book
`int`	`getSoldUnits()` Retrieves number of sold units of the current Book
`java.lang.String`	`getTitle()` Retrieves title of current Book
`int`	`getYear()` Retrieves year when current Book was published
`java.lang.String`	`toString()` Returns a representation of the current Book in String form

Implement a program class that:

a) (0.5 points) Reads from keyboard the ISBN for two books and creates the corresponding `Book` objects

b) (2.5 points) Checks if the two books have the same title but with different year publication, and in that case, shows the message "The modern version is (shorter/equal length/longer) than the old version", where "(shorter/equal length/longer)" must be according to the number of pages of the versions

c) (2 points) Checks if the two books have the same title but with different editorial, and in that case, shows the message "EDITORIAL was more successful", according to the sold units of the book; in case the same units were sold, no message must be printed

d) (2 points) In any case, shows on the screen the book data (returned by the `toString` method) of the book with highest average sales by year until current year 2016 (you can suppose all books will have a publication year previous to that); in case average sales are the same, the two books must be shown

```
import java.util.*;

public class BookSalesManager {
  public static void main(String [] args) {
    Scanner kbd=new Scanner(System.in).useLocale(Locale.US);
    String isbn1, isbn2;
    double avgSales1, avgSales2;
    Book b1, b2;

    System.out.print("First book ISBN: "); isbn1 = kbd.nextLine();
    b1 = new Book(isbn1);

    System.out.print("Second book ISBN: "); isbn2 = kbd.nextLine();
    b2 = new Book(isbn2);

    if (b1.getTitle().equals(b2.getTitle()) && b1.getYear()!=b2.getYear()) {
      if (b1.getYear()>b2.getYear()) {
        if (b1.getNumPages()>b2.getNumPages())
           System.out.println("The modern version is longer than the old version");
        else if (b1.getNumPages()<b2.getNumPages())
           System.out.println("The modern version is shorter than the old version");
        else
```

```
           System.out.println("The modern version is equal length than the old version");
      } else {
        if (b2.getNumPages()>b1.getNumPages())
           System.out.println("The modern version is longer than the old version");
        else if (b2.getNumPages()<b1.getNumPages())
           System.out.println("The modern version is shorter than the old version");
        else
           System.out.println("The modern version is equal length than the old version");
      }
    }

    if (b1.getTitle().equals(b2.getTitle()) && !b1.getEditorial().equals(b2.getEditorial()))
      if (b1.getSoldUnits()>b2.getSoldUnits())
        System.out.println(b1.getEditorial() + " was more successful");
      else if (b2.getSoldUnits()>b1.getSoldUnits())
        System.out.println(b2.getEditorial() + " was more successful");

    avgSales1 = (double) b1.getSoldUnits() / (2016-b1.getYear());
    avgSales2 = (double) b2.getSoldUnits() / (2016-b2.getYear());

    if (avgSales1>avgSales2) System.out.println(b1);
    else if (avgSales2>avgSales1) System.out.println(b2);
    else System.out.println(b1+"\n"+b2);
  }
}
```