Practice Assignment 6

Model Solutions

Exercise 6–1 Airplane
To be discussed in the tutorials

An airplane model can be described (highly oversimplified) by the following attributes

- name: `String`
- empty weight: `double`
- number of seats: `int`
- fuel Consumption: `double`

(a) Define a class `AirplaneModel` with the attributes defined above.

(b) Augment your class with a constructor that initializes an airplane with a name, empty weight, number of seats, and fuel Consumption.

(c) Augment your class with the following methods:

- `getName()` to get the model's name
- `getEmptyWeight()` to get the model's empty weight
- `getSeats()` to get the number of seats
- `getFuelConsumption()` to get the fuel consumption
- `AddSeats(int seats)`: to add `seats` to an airplane.
- `toString()` to display a full report about the airplane including name, empty weight, number of seats, and fuel Consumption.
- Augment your class with a method that returns the total number of airplanes.

(d) Augment your class with a method `main` that constructs one airplane model and performs the following

- Initialize one airplane model. The airplane has `200` seats.
- Display the number of seats of the airplane.
- Add 50 seats to the airplane
- Display the number of seats of the airplane.
- Display a report about the airplane.

Solution:

```
public class AirplaneModel {
  private final String name;             // model name
  private final double emptyWeight;      // model empty weight
  private final int seats;               // model number of seats
  private final double fuelConsumption; // model fuel consumption

  public AirplaneModel(String name, double emptyWeight,
                       int seats, double fuelConsumption) {
    this.name = name;
    this.emptyWeight = emptyWeight;
```

```
    this.seats = seats;
    this.fuelConsumption = fuelConsumption;
  }

  public String getName() {
    return name;
  }

  public double getEmptyWeight() {
    return emptyWeight;
  }

  public int getSeats() {
    return seats;
  }

  public double getFuelConsumption() {
    return fuelConsumption;
  }
  public String toString() {
    return name+"[ew="+emptyWeight+",s="+seats+
      ",fc="+fuelConsumption+"]";
  }
   public static void main(String[] args)
   {
       AirplaneModel airplane = new AirplaneModel("sl", 1.0, 400, 2.0);
       System.out.println( airplane.toString());
   }
}
```

Exercise 6–2 Time
To be discussed in the lab

Write a class named `Time` with the following attributes:

- Hours (`int`)
- Minutes (`int`)
- Seconds (`int`)

(a) Define a class `Time` with the attributes defined above.

(b) Augment your class with two constructors. The first constructor takes no arguments, and sets the hours, the minutes and the seconds to 0. The overloaded constructor takes the three arguments, hours, minutes and seconds.

(c) Augment your class with the following methods:

- A method `hours()` that returns the hours.
- A method `minutes()` that returns the minutes.
- A method `seconds()` that returns the seconds.
- A method `addminute()` that will add one minute to the time and returns the new time.
- A method `addsecond()` that will add one second to the time and returns the new time.
- A method `addhour()` that will add one hour to the time and returns the new time.
- A method `Compare(Time t)` that returns `true` if and only if the time on which the method will be invoked comes after the time `t`.
- A method `toString()` to display a full report about a time (showing of course all attribute values).

(d) Augment your class with a method `main` that constructs at least two `Time` objects and tests all methods defined above.

Solution:

```
import java.util.*;
class Time {
  int hour;
  int min;
  int sec;
  static int count=0;

  Time(int h,int m,int s)
  {
    hour=h;
    min=m;
    sec=s;
    count++;
  }

  Time()
  {
    hour=0;
    min=0;
    sec=0;
    count++;
  }

  public int hours()
  {
    return hour;
  }

  public int minutes()
  {
    return min;
  }

  public int seconds()
  {
    return sec;
  }

  public Time addhour()
  {
    if(hour == 23)
      hour = 0;
    else
        hour++;

    return this;
  }

  public Time addminute()
  {
    if(min == 59)
     {
      min = 0;
      this.addhour();
     }
    else min++;
    return this;
  }
```

```
  public Time addsecond()
  {
    if(sec == 59)
    {
      sec = 0;
      this.addminute();
      if(min == 60)
      {
        min = 0;
        this.addhour();
      }
    }
    else sec++;
    return this;
  }

  public boolean Compare(Time t)
  {
    if(hour > t.hour)
      return true;
    else if(hour == t.hour)
    {
       if(min > t.min)
        return true;
       else if(min == t.min)
       {
        if(sec > t.sec)
           return true;
        else
          return false;
       }
         else return false;
    }
    else return false;
  }
  public static getNumber()
  {
    return count;
  }
  public String toString()
  {
    return "Time now is: " + hour + ":" + min + ":" + sec + "\n";
  }
  public static void main(String[] args)
    {
      Time t = new Time(23, 59, 59);
      Time r = new Time(22, 59, 59);
      System.out.println(t);
      System.out.println(r);
      System.out.println(t.Compare(r));
      System.out.println(t.addhour());
      System.out.println(t.addminute());
      System.out.println(t.addsecond());
   }
}
```

Exercise 6–3 Politics
To be discussed in the tutorials

Develop a class `Country` with the following attributes:

- CountryName : `String`
- noOfCitizens : `int`
- isRoyal : `boolean`
- continent : `String`
- politicalState (4=peace, 3=increase intelligence, 2=Increase in force readiness,1=war): `int`

(a) Define a class **Country** with the attributes defined above.

(b) Augment your class with a constructor that intitalizes a country with a country name, number of citizens, whether it is a royal or not, the name of the continent and the politicalState.

(c) Augment your class with the following methods:

- `getState()` to get the political state.
- `getRoyalState()` to get whether this country is royal.
- `setDefCon(int p)` to set the political state of the country.
- `increaseCitizen(int c)` to increase the number of citizens by a given value.
- `compareTo(Country a)` returns 0 if the number of citizens on which the method will be invoked is the same as the number of citizens of country `a`. It returns a positive number if the country on which the method will be invoked as a larger number than the number of citizen of `a`. And if it has less number of citizens that `a` then a negative number should be returned.
- `toString()` to display a full report about a country (showing of course all attribute values).

(d) Augment your class with a method `main` that constructs movie and performs the following

- Construct a country and initialize it.
- Display a full report about the country.
- Set the political state of the country.
- Construct a second country and initialize it. Display the total number of created countries using the method defined above.
- Compare the number of citizens in the two countries.

Solution:

```
public class Country {

  String  countryName;
  int   noOfCirizens;
  boolean isRoyal;
  String  continent;
  int   politicalState;
  static int count = 0;

  public Country() {
    countryName   = "";
    noOfCirizens  = 0;
    isRoyal     = false;
    continent     = "";
    politicalState  = 0;
    count++;
  }

  public Country(String n, int nc, boolean r,
```

```
        String c, int p) {

  countryName   = n;
  noOfCirizens  = nc;
  isRoyal     = r;
  continent     = c;
  politicalState  = p;
  count++;
}

public String getPoliticalState() {
  String result;
  switch(politicalState) {
    case 1:
      result = "war";
      break;
    case 2:
      result = "Increase␣in␣force␣readiness";
      break;
    case 3:
      result = "increase␣intelligence";
      break;
    case 4:
      result = "peace";
      break;
    default:
      result = "not␣entered␣yet";
  }
  return result;
}
public int compareTo(Country a){
  return noOfCirizens - a.noOfCirizens;
}
public boolean isKingdom() {
  return isRoyal;
}
public void setPoliticalState(int p) {
  politicalState = p;
}
public void increaseCitizens(int i) {
  noOfCirizens += i;
}
public boolean hasMoreCitizens(Country c) {
  return (c.noOfCirizens > this.noOfCirizens);
}
public String toString() {
  return "Country␣name␣" + countryName + "\n"
      + "Its␣population␣is␣" + noOfCirizens + "\n"
      + "It␣" + (isRoyal ? "is␣" : "isn't␣") +"a␣Kingdom\n"
      + "It␣is␣located␣in␣" + continent + "\n"
      + "Its␣political␣state␣is␣" + getPoliticalState();
}
public static int getCountriesCount() {
  return count;
}
public static void main (String [] args) {
  Country    c1   = new Country();
  c1.countryName  = "Egypt";
  c1.isRoyal    = false;
  c1.continent  = "Afrika";
  Country    c2 = new Country("Germany", 20, false, "Europe", 4);

  System.out.println(c1);
```

```
    System.out.println(c2);

    c1.politicalState = c2. politicalState;

    System.out.println(c1);
    System.out.println(c1.hasMoreCitizens(c2));
    System.out.println("Total␣number␣of␣countries␣created␣is␣" + getCountriesCount() );
  }
}
```

Exercise 6–4 Clerk
To be discussed in the tutorial

A clerk is given by the following attributes:

- his Name and lastname (of type `String`)
- the year of birth (of type `int`)
- his salary class (of type `int`)
- a boolean value that indicates whether the clerk is married.

The purpose of the assignment is to design a class for clerks and to find out his monthly salary given the information above. The monthly salary is calculated as follows:

The base salary of 2000 Euro will be increased by $x \cdot 17/9$ Percent, where x is the salary class. Starting with the year, in which the clerk will be 30 years old, the calculated amount increases each 5 years by 1 Percent The allowance for married clerks is 12.3 percent from the base salary.

(a) Define a class `Clerk` with the attributes defined above.

(b) Augment your class with a constructor `Clerk(String firstName, String lastName, int yearOfBirth, int salaryClass, boolean married)`. This constructor has to initialise the instance variables of a clerk with the actual parameters.

(c) Augment your class with a method `void promote(int newSalaryClass)` that will classify a clerk into a new salary class.

(d) Augment your class with the methods `void marry()` and `void divorce()`, that change the marital status (married or not married).

(e) Augment your class with a method `double salary(int thisYear)` that calculates the salary. In the variable `thisYear` the year, in which the salary has to be calculated, will be stored.

(f) Augment your class with a method `main` that constructs two clerks and performs the following

- Display the salary of both clerks.
- The first clerk get married. Then display the salary of both clerks
- The second clerk change to the salary class 7.
- Display whether the first clerk envies the second clerk.

Solution:

```
public class Clerk
{
  /* (a): Instance variables */
  private String firstName, lastName = null;
  private int yearOfBirth = 0;
  private int salaryClass = 0;
  private boolean married = false;

  private final double basicSalary = 2000.0;
  private final double classMultiplier = 17.0 /  9.0;
  private final double marriageBonus = 12.3;

  /* (b):  Constructor */
  public Clerk (String firstName, String lastName, int yearOfBirth,
                int salaryClass, boolean married)
  {
    this.firstName   = firstName;
    this.lastName    = lastName;
    this.yearOfBirth = yearOfBirth;
    this.salaryClass = salaryClass;
    this.married   = married;
  }
 /* (c): Promote a clerk  */
 public void promote(int newSalaryClass)
 {
  salaryClass = newSalaryClass;
 }

 /* (d): Change of marital status */
 public void marry()
 {
  married = true;
 }

 public void divorce()
 {
  married = false;
 }

 /* (e): Calculating the salary  */
 public double salary(int thisYear) {
     double thesalary;
     thesalary = basicSalary * ((100.0 + classMultiplier) / 100.0 * salaryClass);
     System.out.println("s1: " + thesalary);

     /* Age Bonus  */
     int age = thisYear - yearOfBirth;
     int nbonus = 0;
     if (age >= 30)
      nbonus = (age - 25) / 5;

     thesalary = ((100.0 + nbonus) / 100.0) *  thesalary;

     /* Marriage Bonus */
     if (married)
       thesalary += ((100.0 + marriageBonus) / 100.0) * basicSalary;
     return thesalary;
  }

  /*  (g): main: More than required! */
  public static void main(String[] args) {
      Clerk c1 = new Clerk("Mohammed", "Abbas", 1970, 1, false);
      Clerk c2 = new Clerk("John", "Shoukry", 1980, 1, false);
```

```
        System.out.println("c1 earns " + c1.salary(2003));
        System.out.println("c2 earns " + c2.salary(2003));
        c2.promote(7);
        System.out.println("c2 earns " + c2.salary(2003));
    }
}
```

Exercise 6–5 Complex Numbers
To be discussed in the lab

In this exercise, we will develop a complex number class.

A complex number needs two memory locations reserved for the real and imaginary parts and it must provide methods to perform operations such as addition and subtraction.

So the complex class will possess two floating point fields for the real and imaginary values and several methods to carry out the operations allowed on these values.

(a) Define a class `Complex` with the attributes defined above.

(b) Augment your class with a constructor that initializes the values.

(c) Augment your class with the following methods:

- A method to get real and imaginary parts
- Two methods to add two complex numbers. The first method has only one parameter as input:

  `public void add(Complex cvalue)`

  The second method has two parameters as input (the method should create a new Complex object with the sum):

  `public static Complex add(Complex cvalue1, Complex cvalue2)`
- Two methods to subtract a complex number from another complex number.
- A method to display the real and imaginary parts in the following form

  `4.1 + 3.9i`

(d) A `main` method to test your class.

Solution:

```
public class Complex{
  double r;
  double i;

  public Complex() {
    this.r = 0;
    this.i = 0;
  }
  public Complex(double r, double i) {
    this.r = r;
    this.i = i;
  }
  public double getReal() {
    return this.r;
  }
  public double getImaginary() {
    return this.i;
```

```
  }
  public String toString() {
    return "" + this.r + "␣+␣" + this.i + "␣i";
  }
  public static Complex add(Complex c, Complex d) {
    return new Complex(c.r + d.r, c.i + d.i);
  }
  public void add(Complex c) {
    this.r += c.r;
    this.i += c.i;
  }
  public static Complex subtract(Complex c, Complex d) {
    return new Complex(c.r - d.r, c.i - d.i);
  }
  public void subtract(Complex c) {
    this.r -= c.r;
    this.i -= c.i;
  }
  public boolean isEqual(Complex c) {
    return ((this.i == c.i) && (this.r == c.r));
  }
  public static void main (String args []) {
    Complex c1 = new Complex(4, 3);
    Complex c2 = new Complex(5, 2);
    Complex c3 = add (c1, c2);
    System.out.println(c3);
    c3.add(c1);
    System.out.println(c3);
    System.out.println(c2.isEqual(c3));
  }
}
```