COMP 248 **Assignment 3** **Summer 2015**

Deadline:	Monday June 15, 2015, at 11:59pm.
Evaluation:	8% of final mark
Late Submission:	none accepted
Teams:	The assignment can be done individually or in teams of 2. Submit only one assignment per team.
Purpose:	The purpose of this assignment is to help you practice arrays (program 1) and OOP (program 2).

In this assignment, you will write 2 programs:

Program 1: Game of Kenjo-Chima

Write a program to simulate a game of **Kenjo-Chima** and display statistics about the outcome of 1000 games. The game of **Kenjo-Chima** is a single-player dice game. The payer rolls a pair of dice and depending on the outdome of the dice, he/she wins or losses. Here are the rules of the game.

1. The player rolls two standard dice. Their values are added up and called the *Kenjo*.
2. If the *Kenjo* is 2, 3, or 12, the player frowns and loses 7$.
3. Otherwise:
   i. If the Kenjo is 7 or 11, the player screams *Kenjo* and wins 10$.
   ii. Otherwise:
      1. He rolls the two dice again. Their values are added up and called the *Chima.*
      2. If the *Chima* is equals to the *Kenjo*, the player screams *Kenjo-Chima* and wins 5$.
      3. Otherwise:
         a. If the *Chima* equals 7, the player keeps quiet and loses 7$.
         b. Otherwise the player rolls again until he/she gets the *Kenjo* (see rule 3.B.b) or the value 7 (see rule 3.B.c.i).

Your program should simulate 1000 games of Kenjo-Chima and keep track of the 1000 values of the Kenjo, the 1000 values of the last Chima in 2 different arrays. Once the simulation of the 1000 games is done, your program should display:

1. The values of the first 25 Kenjo and first 25 Chima.
2. The values of the last 25 Kenjo and last 25 Chima.
3. The number of times the player frowned (see rule 2 above).
4. The percentage of times the Kenjo had a value of 2, 3, … 11, 12. (use an array for this).
5. The percentage of times the Chima had a value of 2, 3, … 11, 12. (use another array for this).
6. The total amount of money won (or lost) by the player.

Note that if the Chima is not applicable because rule 2 above applies, then consider that the value of the Chima is zero. For example, if the Kenjo is equal to 2, 3, or 12, then there is no need to roll the dice again, and hence there will be no Chima. In that case, consider that the the Chima actually has a value of zero.

Note 2: The file `SampleToGenerateRandomIntegers.java` (available on Moodle) shows how to generate a random integer between 1 and 6. Feel free to use this to simulate the throwing of the dice..

Here is an example of how your program should behave:

```
Welcome to the Kenjo-Chima simulation. Let’s run a simulation of 1000 games

1.      The first 25 Kenjo values are: 8 2 7 4 7 9 9 10 10 8 5 4 6 9 4 6 7 5 7 9 5 8 8 3 7
        The first 25 Chima values are: 7 0 0 7 0 7 7 7 7 7 5 7 7 7 7 7 0 7 0 7 5 8 8 0 0

2.      The last 25 Kenjo values are: 9 7 8 4 3 12 7 12 8 9 10 6 4 8 8 4 9 8 11 8 3 4 6 10 8
        The last 25 Chima values are: 7 0 8 4 0 0 0 0 7 9 7 7 4 8 8 7 7 7 0 8 0 7 6 7 7

3.      The player frowned 115 times

4.      The Kenjo was equal to 2 3% of the time
        The Kenjo was equal to 3 5% of the time
        The Kenjo was equal to 4 8% of the time
        The Kenjo was equal to 5 11% of the time
        The Kenjo was equal to 6 14% of the time
        The Kenjo was equal to 7 13% of the time
        The Kenjo was equal to 8 15% of the time
        The Kenjo was equal to 9 12% of the time
        The Kenjo was equal to 10 8% of the time
        The Kenjo was equal to 11 3% of the time
        The Kenjo was equal to 12 3% of the time

5.      The Chima was equal to 0 28% of the time
        The Chima was equal to 2 0% of the time
        The Chima was equal to 3 0% of the time
        The Chima was equal to 4 3% of the time
        The Chima was equal to 5 4% of the time
        The Chima was equal to 6 6% of the time
        The Chima was equal to 7 43% of the time
        The Chima was equal to 8 7% of the time
        The Chima was equal to 9 4% of the time
        The Chima was equal to 10 2% of the time
        The Chima was equal to 11 0% of the time
        The Chima was equal to 12 0% of the time

    7.  The player lost 699$

Better luck next time!
```

Program 2: On Our Way to Monopoly

Write a program to simulate the selling of properties on a street. Assume that the street currently has 10 properties to sell. Your job is to keep track of the specifics of these properties and try to sell them. (If you have time and feel like it, this assignment could be used as a base to simulate a game of monopoly – but you are not required to do so in this assignment!)

Your program will consist in 4 classes: `Property`, `Player`, `Monopoly` and `MonopolyDriver`. `Monopoly` and `MonopolyDriver` are give to you (available on Moodle), so your task is be to write the classes `Property` and `Player` so that they run with the given `Monopoly` and `MonopolyDriver` on Moodle and satisfy the following specifications:

The class `Property` must contain at least 3 attributes:

1. the value of the property (e.g. 250,000$)
2. the name of the property (e.g. `“Pennsylvania”` or just `“Property-1”)`
3. the owner of the property – either **`null`** if the property does not belong to anyone; or a reference to a `Person` object if the property belongs to that person.

In addition, the `Property` class must have at least the following methods:

1. a constructor that takes 3 arguments (one for each attribute)
2. a copy constructor that takes another `Property` object as argument and returns a copy of it
3. an appropriate `toString` method
4. a method **`equals`** that returns true if all attributes have the same value.
5. a method called `isSmallerThan` which returns true of the calling property has a value smaller than the property passed as parameter. For example, a property worth 150,000$ is considered smaller than one with a value of 185,000$.
6. a method called `sell` that tries to sell the property to a person. The method `sell` should take a `Person` as parameter and if the person has enough money, then the property’s owner should be changed to this new person, and his/her budget should be decreased accordingly. If the person does not have enough money, then the owner of the property should not change, and the sale should not take place. Your method should display an appropriate message to indicate if the sale took place or not, and indicate the player’s budget before and after the transaction (see the example trace on the next page).
7. A method called `getOwner` that returns the owner of the property (an object of the class Player) or `null` if the property has no owner.

The class `Player` must contain at least 2 attributes:

1. a name (e.g. `“NewYorkGazillionaire”` or just `“Player-1”)`
2. the budget that the player has (e.g. 150,000$)

In addition, the `Player` class must have at least the following methods:

1. a constructor that takes 2 arguments (1 for each attribute)
2. a copy constructor that takes another `Player` object as argument and returns a copy of it
3. an appropriate `toString` method
4. appropriate accessor methods
5. a mutator for the budget. This method should allow a player to change the value of the budget only if it will remain non-negative, otherwise, no change should occur.

Once your two classes are complete, make sure that they run with the classes `Monopoly` and `MonopolyDriver` given to you, and produce results similar to the following:

```
Welcome to my Monopoly Game!

Here are 10 random properties:
Property-0 value: $4,806.00 Owner: null
Property-1 value: $90,394.00 Owner: null
Property-2 value: $60,403.00 Owner: null
Property-3 value: $76,132.00 Owner: null
Property-4 value: $26,597.00 Owner: null
Property-5 value: $19,174.00 Owner: null
Property-6 value: $28,186.00 Owner: null
Property-7 value: $30,210.00 Owner: null
Property-8 value: $47,578.00 Owner: null
Property-9 value: $90,630.00 Owner: null

The properties sorted by their values:
Property-0 value: $4,806.00 Owner: null
Property-5 value: $19,174.00 Owner: null
Property-4 value: $26,597.00 Owner: null
Property-6 value: $28,186.00 Owner: null
Property-7 value: $30,210.00 Owner: null
Property-8 value: $47,578.00 Owner: null
Property-2 value: $60,403.00 Owner: null
Property-3 value: $76,132.00 Owner: null
Property-1 value: $90,394.00 Owner: null
Property-9 value: $90,630.00 Owner: null

Let's try to sell the properties to random players:
Property-9 value is $90,630.00 but Player-0 only has $73,790.00 --> NOT sold! Player-0 has $73,790.00
Property-3 value is $76,132.00 and Player-1 has $92,695.00 --> SOLD! Player-1 has $16,563.00
Property-8 value is $47,578.00 and Player-3 has $65,922.00 --> SOLD! Player-3 has $18,344.00
Property-6 value is $28,186.00 and Player-4 has $46,696.00 --> SOLD! Player-4 has $18,510.00
Property-5 value is $19,174.00 but Player-6 only has $6,094.00 --> NOT sold! Player-6 has $6,094.00
Property-0 value is $4,806.00 and Player-7 has $85,327.00 --> SOLD! Player-7 has $80,521.00
Property-5 value is $19,174.00 and Player-8 has $33,135.00 --> SOLD! Player-8 has $13,961.00

The properties with their new owners:
Property-0 value: $4,806.00 Owner: <Player-7 with $80,521.00>
Property-5 value: $19,174.00 Owner: <Player-8 with $13,961.00>
Property-4 value: $26,597.00 Owner: null
Property-6 value: $28,186.00 Owner: <Player-4 with $18,510.00>
Property-7 value: $30,210.00 Owner: null
Property-8 value: $47,578.00 Owner: <Player-3 with $18,344.00>
Property-2 value: $60,403.00 Owner: null
Property-3 value: $76,132.00 Owner: <Player-1 with $16,563.00>
Property-1 value: $90,394.00 Owner: null
Property-9 value: $90,630.00 Owner: null

Thanks for playing!
```

Evaluation Criteria for Program 1 (out of 60):

Comments: 4 pts	
Description of the program (authors, date, purpose):	1 pt
Description of variables and constants:	1 pt
Description of the algorithm:	2 pts
Programming style: 14 pts	
Use of constants where appropriate:	2 pts
Use of significant names for identifiers:	2 pts
Indentation and readability:	4 pts
Simplicity of the overall program:	6 pts
Functionality: 42 pts	
Values of the first 25 Kenjo and first 25 Chima	8 pts
Values of the last 25 Kenjo and last 25 Chima	8 pts
Money won (or lost) by the player	5 pts
Number of times the player frowned	5 pts
Percentage of times the Kenjo fell on each value	8 pts
Percentage of times the Chima fell on each value	8 pts

Evaluation Criteria for Program 2 (out of 40):

Comments: 3 pts	
Description of the program	1 pt
Description of variables, methods…	2 pts
Programming style: 4 pts	
Use of significant names for identifiers:	2 pts
Indentation and readability:	2 pts
Implementation of the class **`Property:`** 18 pts	
Declaration of the attributes	2 pts
Constructor with 3 arguments	2 pts
Copy constructor	3 pts
`toString` method	2 pts
`equals` method	2 pts
`isSmallerThan` method	1 pts
`sell` method	3 pts
`getOwner` method	2 pts
Implementation of the class **`Player:`** 15 pts	
Declaration of the attributes	2 pts
Constructor with 2 arguments	2 pts
Copy constructor	3 pts
`toString` method	2 pts
Accessor methods	3 pts
Mutator for the budget	3 pts

Demonstrations: In addition to the evaluation criteria above, please note that part of the submission may involve a short 10 minute demonstration during the lab period. If your team is called for a demo, all members of the group must attend the demo.
No extra preparation is necessary – you only need to explain your code to the lab instructor and answer his/her questions. All members of the group must be able to explain their program to the marker. Different marks may be assigned to teammates based on this demo. The list of teams that are called for a demo and the specific schedule of the demos will be announced on the Moodle page. Failure to do your demo will entail a mark for zero for the assignment regardless of the code submitted.

General Guidelines When Writing Programs:

Identical to the previous assignments.

Submission

When you are finished the programs, you must submit tem online.

1) Create **one** zip file, containing all necessary source files (the .java files).

Please name your file following this convention:

- If the work is done by 1 student, your file should be called *a#_studentID*, where # is the number of the assignment *studentID* is your student ID number. For example, for the first assignment, student 12345678 should submit: `a3_12345678.zip`
- If the work is done by 2 students: The zip file should be called *a#_studentID1_studentID2*, where # is the number of the assignment *studentID1* and *studentID2* are the student ID numbers of each student, for example, `a3_12345678_87654321.zip`
  Only one member of the team need to upload the file.

2) Upload your zip file at the URL: https://fis.encs.concordia.ca/eas/ as *Programming Assignment 3*