Practical Software Engineering I.

Exercises for the 1st assignments

Common requirements:

- Use a Collection to store the objects of classes derived from the same super class.
- Use foreach to process the elements of a Collection.
- Validate the data what you get from the user; throw Exception for invalid data, and handle the thrown Exceptions.
- The documentation should contain:
    - the description of the exercise,
    - the class diagram,
    - the short description of each methods,
    - and the testing (white box / black box).

1. There is a race for creatures, which takes place on several consecutive days. ***Who wins the race? (In other words, which creature can go farthest and remain live?)*** At the beginning, each creature has an amount of water, and a distance of 0 from the start. There are three different kind days could occur: sunny, cloudy, rainy. The movement and the water level of a creature are affected by the type of the day and the creature. At first, a creature changes its water level according to the day, and if it is still alive, it moves. A creature dies if it runs out of water (water level drops to 0 or below). A dead creature doesn't move…
   Properties of creatures: name of the creature (string), water level (integer), maximum water level (integer), living (boolean), distance (integer).
   The types of creatures on the race are: sandrunner, sponge, walker.
   The following table contains the properties of the creatures.

	water change			**distance**			**max. water**
	sunny	cloudy	rainy	sunny	cloudy	rainy	
sandrunner	-1	0	3	3	1	0	8
sponge	-4	-1	6	0	1	3	20
walker	-2	-1	3	1	2	1	12

   Creatures cannot have water more than their maximum water level.
   Read the data of the race from a text file. The first line of the file contains the number of competitors (lets say N). Each of the following N lines contains a competitor: name, type, initial water level. The properties are separated by spaces; and the type is represented with one character: r - sandrunner, s - sponge, w - walker.
   The last line of the file contains the type of the days on the race: s - sunny, c - cloudy, r - rainy.
   The program should ask for the name of the file, and it has to print out the name of the winner (we can assume that the file is existing and its format is valid).

   A possible file content:

   4
   wanderer r 4
   walk w 7
   slider s 12
   sneaky s 10
   sccrrrssc

   (Tips: Create an abstract Creature class, and derive the three kind of creatures. Let this class have a constructor with the parameters of name and initial water level. Introduce three methods for each days (sunny, cloudy, rainy), which updates the water level, checks the life of the creature, and handles the movement. To obtain the final result, the following 3 methods are also required: isAlive, getName, getDistance.)

2. There is a planet, where different kind of plants are living. All the plants are using nutrients to live. If a plant runs out of its nutrients, it dies. Each day one radiation type can occur from the followings: alpha, delta, or no radiation. Radiations affect the plants differently based on their types. The reaction of a plant to a given radiation consists of the following: it changes its nutrient level, and affects the radiation of the next day. The radiation of the next day:
    a. alpha, if the need for alpha radiation is 3 or more greater than for the delta radiation
    b. delta, if the need for delta radiation is 3 or more greater than for the alpha radiation
    c. no radiation, otherwise

    There is no radiation on the first day…

    Simulate the behaviors of the plants, and print out the radiation of the day and the properties of the plants on each day.

    Properties of the plants: name (string), nutrients (integer), living (boolean). The types of the plants in the simulation: puffs, deltatree, parabush.
    On a day of the the simulation the living plant first changes its nutrients, then if it is still alive, it can affect the radiation of the next day.

	nutrients (N)			radiation need on next day			dies
	alpha	delta	no radiation	alpha	delta	no radiation	
Puffs	+2	-2	-1	10-N			10<N
Deltatree	-3	+4	-1		+4, if N < 5 +1, if 5 ≤ N ≤ 10		
Parabush	+1	+1	-1				

    Read the data of the simulation from a text file. The first line contains the number (n) of the plants. The following n lines contain the information about the plants: name, type, initial nutrient level. Type is represented by one character: p - Puffs, d - Deltratree, b - Parabush. The last line of the file defines the number of the days you have to simulate.
    The program should ask for the name of the file, and it has to print out the name of the survivors (we can assume that the file is existing and its format is valid).

    A possible file content:
    4
    Piggy p 7
    Slender d 5
    Dumpy b 4
    Willowy d 3
    10

3. Simulate a simplified Capitaly game. There are some players with different strategies, and a cyclical board with several fields. Players can move around the board, by moving forward with the amount they rolled with a dice. A field can be a property, service, or lucky field. A property can be bought for 1000, and stepping on it the next time the player can build a house on it for 4000. If a player steps on a property field which is owned by somebody else, the player should pay to the owner 500, if there is no house on the field, or 2000, if there is a house on it. Stepping on a service field, the player should pay to the bank (the amount of money is a parameter of the field). Stepping on a lucky field, the player gets some money (the amount is defined as a parameter of the field). There are three different kind of strategies exist. Initially, every player has 10000.
   Greedy player: If he steps on an unowned property, or his own property without a house, he starts buying it, if he has enough money for it.
   Careful player: he buys in a round only for at most half the amount of his money.
   Tactical player: he skips each second chance when he could buy.
   If a player has to pay, but he runs out of money because of this, he loses. In this case, his properties are lost, and become free to buy.

   Read the parameters of the game from a text file. This file defines the number of fields, and then defines them. We know about all fields: the type. If a field is a service or lucky field, the cost of it is also defined. After the these parameters, the file tells the number of the players, and then enumerates the players with their names and strategies.
   In order to prepare the program for testing, make it possible to the program to read the roll dices from the file.
   Print out what we know about each player after a given number of rounds (balance, owned properties).

4. Simulate a simplified Capitaly game. There are some players with different strategies, and a cyclical board with several fields. Players can move around the board, by moving forward with the amount they rolled with a dice. A field can be a property, service, or lucky field. A property can be bought for 1000, and stepping on it the next time the player can build a house on it for 4000. If a player steps on a property field which is owned by somebody else, the player should pay to the owner 500, if there is no house on the field, or 2000, if there is a house on it. Stepping on a service field, the player should pay to the bank (the amount of money is a parameter of the field). Stepping on a lucky field, the player gets some money (the amount is defined as a parameter of the field). There are three different kind of strategies exist. Initially, every player has 10000.
   Greedy player: If he steps on an unowned property, or his own property without a house, he starts buying it, if he has enough money for it.
   Careful player: he buys in a round only for at most half the amount of his money.
   Tactical player: he skips each second chance when he could buy.
   If a player has to pay, but he runs out of money because of this, he loses. In this case, his properties are lost, and become free to buy.

   Read the parameters of the game from a text file. This file defines the number of fields, and then defines them. We know about all fields: the type. If a field is a service or lucky field, the cost of it is also defined. After the these parameters, the file tells the number of the players, and then enumerates the players with their names and strategies.
   In order to prepare the program for testing, make it possible to the program to read the roll dices from the file.
   Print out which player won the game, and how rich he is (balance, owned properties).

5. Simulate a simplified Capitaly game. There are some players with different strategies, and a cyclical board with several fields. Players can move around the board, by moving forward with the amount they rolled with a dice. A field can be a property, service, or lucky field. A property can be bought for 1000, and stepping on it the next time the player can build a house on it for 4000. If a player steps on a property field which is owned by somebody else, the player should pay to the owner 500, if there is no house on the field, or 2000, if there is a house on it. Stepping on a service field, the player should pay to the bank (the amount of money is a parameter of the field). Stepping on a lucky field, the player gets some money (the amount is defined as a parameter of the field). There are three different kind of strategies exist. Initially, every player has 10000.
   Greedy player: If he steps on an unowned property, or his own property without a house, he starts buying it, if he has enough money for it.
   Careful player: he buys in a round only for at most half the amount of his money.
   Tactical player: he skips each second chance when he could buy.
   If a player has to pay, but he runs out of money because of this, he loses. In this case, his properties are lost, and become free to buy.

   Read the parameters of the game from a text file. This file defines the number of fields, and then defines them. We know about all fields: the type. If a field is a service or lucky field, the cost of it is also defined. After the these parameters, the file tells the number of the players, and then enumerates the players with their names and strategies.
   In order to prepare the program for testing, make it possible to the program to read the roll dices from the file.

   Print out which player loses as a second loser.

6. Choose a point on the plane, and fill a collection with several regular shapes (circle, regular triangle, square, regular hexagon). ***How many shapes contain the given point?***

   Each shape can be represented by its center and side length (or radius), if we assume that one side of the polygons are parallel with x axis, and its nodes lies on or above this side. Load and create the shapes from a text file. The first line of the file contains the number of the shapes, and each following line contain a shape. The first character will identify the type of the shape, which is followed by the center coordinate and the side length or radius. Manage the shapes uniformly, so derive them from the same super class.

7. Fill a collection with several regular shapes (circle, regular triangle, square, regular hexagon). ***Determine the smallest bounding box, which contains all the shapes, and its sides parallel with an x or y axis.***

   Each shape can be represented by its center and side length (or radius), if we assume that one side of the polygons are parallel with x axis, and its nodes lies on or above this side. Load and create the shapes from a text file. The first line of the file contains the number of the shapes, and each following line contain a shape. The first character will identify the type of the shape, which is followed by the center coordinate and the side length or radius. Manage the shapes uniformly, so derive them from the same super class.

8. Choose a point on the plane, and fill a collection with several regular shapes (circle, regular triangle, square, regular hexagon). ***Which shapes lies the closest to the point?***

   In case of the circle, the distance is measured from the line of the circle, if the point lies outside of the circle. If a point lies inside of a circle, their distance is considered as zero.
   Each shape can be represented by its center and side length (or radius), if we assume that one side of the polygons are parallel with x axis, and its nodes lies on or above this side. Load and create the shapes from a text file. The first line of the file contains the number of the shapes, and each following line contain a shape. The first character will identify the type of the shape, which is followed by the center coordinate and the side length or radius. Manage the shapes uniformly, so derive them from the same super class.

9. Fill a collection with several regular shapes (circle, regular triangle, square, regular hexagon). ***Which shape has the smallest difference between its area and perimeter?***

   Each shape can be represented by its center and side length (or radius), if we assume that one side of the polygons are parallel with x axis, and its nodes lies on or above this side. Load and create the shapes from a text file. The first line of the file contains the number of the shapes, and each following line contain a shape. The first character will identify the type of the shape, which is followed by the center coordinate and the side length or radius. Manage the shapes uniformly, so derive them from the same super class.

10. Fill a collection with several regular shapes (circle, regular triangle, square, regular hexagon). ***Which shape has the greatest bounding box area?***

    A bounding box of a shape covers the shape completely, and its sides are parallel with the x or y axis. Each shape can be represented by its center and side length (or radius), if we assume that one side of the polygons are parallel with x axis, and its nodes lies on or above this side. Load and create the shapes from a text file. The first line of the file contains the number of the shapes, and each following line contain a shape. The first character will identify the type of the shape, which is followed by the center coordinate and the side length or radius. Manage the shapes uniformly, so derive them from the same super class.