LP24 Object Oriented Programing
Final Exam - Spring 2015

Documents, Computer, Calculator unauthorized

1. Theoretical questions (4 pts)

A. What are the four pillars of OOP? Explain them briefly.

B. What is the purpose of interfaces? How to write a class which implements an interface in java? How many interfaces a class can implement? Can an interface inherits from a class? from another interface?

C. How does inheritance promote software reusability?

2. Exercise #1: May the force be with you (10 pts)

The code in annexe corresponds to a battle between X-Wings and Tie-Fighters. The X-wing are the goodies and the Tie Fighter are the baddies (reverse this sentence if you are from the dark side). The goal of this exercice is to program the missing classes so as to make this code working properly.

A. Considering the StarWars.java file provided, the compiler will make an error. Correct it. Does it sound good for you if some methods are public? Choose the correct modifier.

B. We now want to build up the ships. A X-wing is defined with internal attributes which correspond to the shields level, to a weapon level and to a boolean which expresses its status (Dead or Alive). According to the StarWars.java file, what are the required methods for a X-wing? Same question for a TIE fighter if we assume that they share the same attributes?

C. According to the previous answer give the content of the X-wing and the Tie Fighter classes. (A X-wing has an initial shield value of 1000 and a weapon power of 10; A Tie Fighter has got respectively 500 and 20)

The hit function is given by the following code:

```
public void hit(int damage) {
     shields = shields - damage;
     if (shields < 0) {
        System.out.println("BOOM!!!");
        dead = true;
     }
```

D. Having code duplicates in a program is not necessarily a good point. Explain why.

E. Change the code for the X-wing and the TieFighter classes so has to avoid duplicate code.

F. I want to introduce a new weapon which can be called by the following method: `void superHit(int damage)`. Write an interface which could be implemented by any space ship.

G. Using the previous elements define two new ships which can have the new weapon (one in each camp).

H. For the dark side, the use of the superHit can be made if the shield value corresponds to an odd number. The result of the use of the new weapon is an increase by two of the weapon power and a reduction of the shield level of 100. For the Rebels, the result of the use of the super weapon is an increase of the weapon power by five and a shield power equals to 1. This weapon can be used only if the shield power is below 100. Make the necessary modifications so has to integrate the new weapon to the ships.

I. If we want to integrate these new ships into the battlefield, what part of the code should we change to do so in the StarWars class. Do the required modifications. Why it is so simple?

3. Exercise #3: Fun with # (6 pts)

The goal of the exercise is to write the required classes which embedded the methods aimed at writing the following patterns on the screen. Try to do so with as few methods as possible. (One method for each pattern is not the correct answer). Each pattern must have parameters for specifying the size of each pattern and must be stored in memory using a collection of your choice. Some of these patterns are composed of several patterns, try to think in terms of composition of patterns while defining a new one. (Tips: one can try to define operations between patterns)

```
#                   # # # # # # # #      # # # # # # # #                   #
# #                 # # # # # # #          # # # # # # #                 # #
# # #               # # # # # #              # # # # # #               # # #
# # # #             # # # # #                  # # # # #             # # # #
# # # # #           # # # #                      # # # #           # # # # #
# # # # # #         # # #                          # # #         # # # # # #
# # # # # # #       # #                              # #       # # # # # # #
# # # # # # # #     #                                  #     # # # # # # # #
     (a)                  (b)                 (c)                  (d)

# # # # # # #    # # # # # # #   # # # # # # #   # # # # # # #    # # # # # # #
#           #      #                     #         #       #      # #       # #
#           #        #                 #             #   #        #   #   #   #
#           #          #             #                 #          #     #     #
#           #            #         #                 #   #        #   #   #   #
#           #              #     #                 #       #      # #       # #
# # # # # # #    # # # # # # #   # # # # # # #   # # # # # # #    # # # # # # #
     (e)              (f)             (g)             (h)              (i)

# # # # # # # # # # #               #                         #
  # # # # # # # # #               # # #                     # # #
    # # # # # # #               # # # # #                 # # # # #
      # # # # #               # # # # # # #             # # # # # # #
        # # #               # # # # # # # # #         # # # # # # # # #
          #               # # # # # # # # # # #     # # # # # # # # # # #
         (j)                       (k)                # # # # # # # # #
                                                        # # # # # # #
                                                          # # # # #
                                                            # # #
                                                              #
                                                         (l)
```

A. Write a class with the necessary methods aimed at writing a line of # on the screen. This class has, as attributes, the size of the line (i.e. the number of #) and its alignment (left, right, center) and a method draw which print it on the screen.
B. Now, write the class, that corresponds to a triangle (fig. (a)). How can we use the previous class? Can inheritance be used in this case?
C. Write a class multipleCharacterLine which is able to write a line having several # separated by spaces such as in the second line of figure (h) and (i) for instance. How can we use the class defined in A?
D. Write the class CheckBox aimed at printing figure (i) on the screen using the preceding classes.

4. Annexe

StarWars.java

```
class StarWars {
   public void duel(XWing x, Tie
t) {boolean Dead=false;

      while (!Dead) {

     x.hit(t.getWeapon());
     if (x.isDead()) {
            System.out.println("X-
Wing is dead");
            Dead=x.isDead();
         }

     t.hit(x.getWeapon());
 if (t.isDead()) {
           System.out.println("Tie
fighter is dead");
            Dead=t.isDead();
         }
                      }
  }

   public void battle(XWing[]
good, Tie[] evil) {

      int g = 0;
      int e = 0;
      int goodDeaths = 0;
      int evilDeaths = 0;

      while (g < good.length && e
< evil.length) {

System.out.println("battling X-
Wing #" + g + " versus Tie Fighter
#" + e);
         duel(good[g],evil[e]);
         if (good[g].isDead()) {
            g++;
            goodDeaths++;
         }
         if (evil[e].isDead()) {
            e++;
            evilDeaths++;
         }
      }

      int finalGood = good.length
- goodDeaths;
      int finalEvil = evil.length
- evilDeaths;

      System.out.println();
      System.out.println("Battle
Report:\t\tX-Wings\t\tTie
Fighters");

System.out.println("-------------
-----------------------------------
------");
      System.out.println();
      System.out.println("Initial
ships:\t\t" + good.length + "\t\t"
+ evil.length);
      System.out.println();
      System.out.println("Killed
ships:\t\t"  + goodDeaths  + "\t
\t" + evilDeaths);
      System.out.println();
      System.out.println("Final
ships:\t\t"   + finalGood   + "\t
\t" + finalEvil);
      System.out.println();
      if (finalGood > finalEvil) {
         System.out.println("The
rebel alliance is victorious!");
      }
      else {
         System.out.println("The
dark side has conquered!");
      }
      System.out.println();
   }

   private void doStuff() {

      // defines the goodies array
      XWing[] goodies = new
XWing[3];

      // initialises the elements
of the goodies array
      for (int i=0;
i<goodies.length; i++) {
         goodies[i] = new XWing();
      }

      // defines the baddies array
      Tie[] baddies = new Tie[3];

      // initialises the elements
of the baddies array
      for (int i=0;
i<baddies.length; i++) {
         baddies[i] = new Tie();
      }

      battle(goodies,baddies);

   }

   // The main method is the point
of entry into the program...
   public static void
main(String[] args) {
      StarWars me = new
StarWars();
      me.doStuff();
   }

}
```