Exam
DIT948: Introduction to Programming

Date:	2014-11-17
Time	8:30–12:30
Place	Lindholmen
Teacher and examiner	Cezar Ionescu
Questions	0708346886 first visit around 9:30
Results	will be posted within 15 working days
Grades	Pass (G) 50p, Pass with Honors (VG) 80p
Allowed aids	Any book on Java programming is allowed Any English dictionary No electronic translators allowed
Reviewing:	2015-01-19 and 20, 10:00–12:00, Lindholmen Second floor room (note: subject to change, changes will be posted on the course website).

Please observe the following:

- Write in legible English (illegible translates to "no points"!).
- Motivate your answers, and clearly state any assumptions made.
- Code or text copied from the book will not be rewarded. Your own contribution is required.
- Start each task on a new sheet!
- Write on only one side of the paper!
- Before handing in your exam, number and sort the sheets in task order!
- Write your anonymous code and page number on every page!

Not following these instructions will result in the deduction of points!

1. [10pts] Consider the class:

```
package Shapes;

public class Rectangle {
    private  int length;
    private  int width;

    double x;
    double y;

    protected int colour;

    private void move(double dx, double dy) {
        x = x + dx;
        y = y + dy;
    }

    public int getLength() { return length; }
    public int getWidth() { return width; }

    protected int perimeter() { return 2 * (length + width); }

    int getColour() { return colour; }

}
```

Class `TestRectangle` is a subclass of `Rectangle`:

```
 1 package Shapes;
 2
 3 public class TestRectangle extends Rectangle {
 4     public void test() {
 5         System.out.println(length);
 6         System.out.println(width);
 7         System.out.println(x);
 8         System.out.println(y);
 9         System.out.println(colour);
10         move(1.2, -2.3);
11         System.out.println(getLength());
```

```
12          System.out.println(getWidth());
13          System.out.println(perimeter());
14          System.out.println(getColour());
15      }
16 }
```

Which of the lines 5–14 will cause a compilation error and why? How does the answer change if `TestRectangle` is in a different package from `Rectangle` (for example, if we replace the first line with `import Shapes.Rectangle;`)?

2. [10pts] Your boss has written a method that was supposed to return the first index at which two given arrays of the same length are different, and to return `-1` if the arrays are identical.

   For example:

```
a = {1, 2, 3} b = {1, 2, 2}   =>  firstDiff(a, b) = 2
a = {1, 2, 3} b = {1, 1, 4}   =>  firstDiff(a, b) = 1
a = {1, 2, 3} b = {0, 2, 3}   =>  firstDiff(a, b) = 0
a = {1, 2, 3} b = {1, 2, 3}   =>  firstDiff(a, b) = -1
```

   Unfortunately, things didn't really work out that way: in fact, the method doesn't even compile! Still, it's your boss's program, you can't just throw it away and write a new one. Make the changes necessary for the method to compile and return the correct result.

```
public static int firstDiff(int a[], int b[]) {
    for (i = 0; i <= length; i++)
        if (a != b)
            return true;
    return -1;
}
```

3. [10pts]

   Given the following classes:

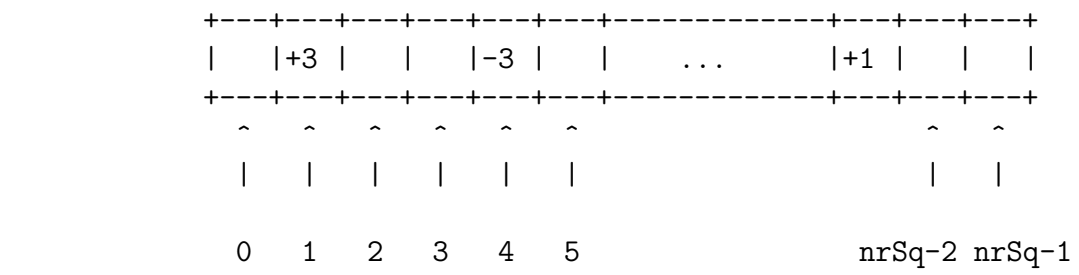
```
public class Vehicle {
    public void drive() {
        System.out.println("Vehicle: drive");
```

```
    }
}

public class Car extends Vehicle {
    public void drive() {
        System.out.println("Car: drive");
    }
}

public class Test {
    public static void main (String args []) {
        Vehicle v;
        Car c;
        v = new Vehicle();
        c = new Car();
        v.drive();
        c.drive();
        c = v;
        c.drive();
        v = c;
        v.drive();
    }
}
```

Which lines do you need to comment out in order to get the class `Test` to compile? Once you do that, what will be the effect of running the class `Test`?

4. [30pts]

Consider the following simple board game. The board is just a long strip divided into squares, numbered sequentially:



The players start at square 0, and roll a dice in order to advance along the strip. Thus, if a player is at the beginning (position 0) and rolls a three, his next position will be 3. If the strip has `nrSq` squares, then the last position is `nrSq-1`.

The strip is considered circular. If a player rolls past the last position, he goes on counting from the beginning. For example, if the player is at position `nrSq-2` (next-to-last square) and he rolls a 3, the next position will be 1. (Subtract `nrSq` from positions that would be larger than `nrSq-1`.)

Each player starts with a score of 0. Some of the squares contain rewards (positive integers) or penalties (negative integers). In the figure above, the squares 1 and `nrSq-3` contain rewards of 3 and 1, respectively. The square 4 contains a penalty of -3.

If a player lands on a such a square, he collects the respective reward or the penalty, i.e. it is added to or subtracted from his score.

The game is played for a given number of rounds. In a round, each player takes one turn rolling the dice, advancing the respective number of squares, and collecting the reward or penalty of the square he lands on.

After the given number of rounds is played, the player with the greatest score wins. Ties are broken in favor of the player who started last.

In the following, you will implement several classes to represent this game. You will do this by "filling in" the data and method definitions. You don't need to do any error checking or write comments.

Do not write "between the lines"! On your paper, clearly mark out which code is supposed to go where. For example, the code supposed to fill in the block `A`, the member variables of the class `Race`, should appear as

```
+----- A -----+
|  int pos;   |
|  int score; |
+-------------+
```

and not squeezed on the page with the exam subjects!

Remarks:

- Since block A has been filled in for you above, there are ten blocks left for you to fill in for this exercise: B, C, D, E, F, G, H, I, J, K. Blocks D and G are two points each, blocks B, C, E, F, H, I, and J are three points each, and block K is 5 points.
- Use the `dit948.Random` method `randomInt` to generate the random numbers needed in blocks C and K.

```
/**
 *  Generates one of n possible results (0, 1, ..., n-1)
 *  from a uniform distribution.
 *
 *  @parameter n the number of results generated
 *  @return a number between 0 and n-1
 */
public static int randomInt(int n);
```

End of remarks.

```
import static dit948.Random.*;

// Class representing the players.
public class Player {
    // Block A:
    // Declare two member variables:
    // pos, an integer representing the player's position
    // score, an integer representing the player's score

    public Player() {
        // Block B:
        // Initialize both the position
        // and the score to 0
    }

    public int rollDice() {
        // Block C:
        // use randomInt to generate a dice roll,
        // i.e., a number between 1 and 6
        // return the result
    }
}
```

```
// Class Square.
public class Square {
    // Block D:
    // Declare here a private member variable
    // of type integer, representing the penalty
    // or reward

    public Square(int n) {
        // Block E:
        // Initialize the member variable
        // with the given argument n
    }

    public int changeScore() {
        // Block F:
        // return the value of the member variable
    }
}

// Class representing the board.
public class Board {
    // Block G:
    // Declare two member variables:
    // squares, an array of Square
    // nrSq, an integer

    public Board(int nrSq) {
        // Block H:
        // Initialize the member variable nrSq
        // with the value given as argument

        // Block I:
        // Allocate the squares array to size nrSq

        // Block J:
        // Initialize the first square with a square
        // that does not change the player's score

        // Block K:
        // Initialize the rest of the squares (from 1
        // to nrSq-1) to squares which change the
```

```
            // player's square with a random value
            // between -9 and 9 (including 0).
    }
```

5. [20pts] In this exercise, we refer to the classes from above, but do not require you to have implemented them (only that you understand how to create and use instances of them).

   (a) Implement a method with the signature

   ```
   public static void round(Board board, Player[] players)
   ```

   representing one round in the game. Each player must roll the dice, advance the resulting number of steps, and collect the respective penalty or reward. You can access the variables of the players and of the board directly, for example, the position of player `i` is `players[i].pos`, and his score is `players[i].score`; similarly, the size of the board is `board.nrSq`, and the square at position `p` is `board.squares[p]`.

   (b) Implement a method with the signature

   ```
   public static int winner(Player[] players)
   ```

   which returns the index of the player with the largest score. Ties are broken in favor of the last player. Thus, if `player[3]` and `player[7]` both have the largest score, the winner is player `7`. As before, you can access the variables of the players directly.

   Each method is worth 10 points.

6. [20pts] (From *CodingBat*)

   Write a subroutine taking an array of `int`s as argument, returning `true` if that array contains three consecutive adjacent numbers, such as `4`, `5`, `6`, or `23`, `24`, `25`. Remember to check the corner cases! The subroutine should always return a value, and not crash.

   Examples:

   ```
   threeConsecutive({1, 4, 5, 6, 2})  ==>  true
   threeConsecutive({1, 2, 3})        ==>  true
   threeConsecutive({1, 2, 4})        ==>  false
   threeConsecutive({1, 2, 4, 5})     ==>  false
   ```