Exam
DIT948: Introduction to Programming

Date:	2015-08-27
Time	8:30–12:30
Place	Lindholmen
Teacher	Cezar Ionescu
Questions	Musard Balliu: 0762416640 first visit around 9:30
Results	will be posted next week
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:	Will be held by Musard Balliu, date to be announced.

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:

```
public class X {
    private int x = 0;

    public X() {}

    public X(int x) {
        this.x = x;
    }

    public void f0() {
        System.out.println("x = " + x);
    }

    public void f1(int x) {
        System.out.println("x = " + x);
    }

    public void f2() {
        int x = 3;
        System.out.println("x = " + x);
    }

    public static void main(String[] args) {
        int x = 1;

        X test = new X(x);

        test.f0();
        test.f1(2);
        test.f2();

        test   = new X();
        test.f0();
        System.out.println("x = " + x);
    }
}
```

What will be printed when running `X`?

2. [10pts] Your boss wrote a program that was supposed to compute the *factorial* of a number `i`, where `i` is given via the static member variable of the class `Factorial`. The factorial of `i` is the product of all numbers from 1 to `i`. If `i` is 0, then the factorial is 1. Thus, if `i = 3`, the result should be `1 * 2 * 3 = 6`, and if `i = 5`, the result should be `1 * 2 * 3 * 4 * 5 = 120`. Unfortunately, things didn't really work out that way: in fact, the program 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 program to compile and print the correct result.

```
public class Factorial {
  static int i = 3;

  public static void main(String[] args) {
    System.out.println("i = " + i);
    for (int i = 0; i < i; i++) {
      System.out.println("i = " + i);
      int fact = 0;
      fact = i * fact;
    }
    System.out.println("The factorial is " + fact);
  }
}
```

3. [10pts]

   Given the following classes:

```
class A {
    int x = 1;

    public String toString() {
        return "In class A " + x;
    }
}

class B extends A {
    int x = 3;

    public String toString() {
        return "In class B " + x;
    }
}

class Test {
    public static void main(String[] args) {
        A a1 = new A();
        System.out.println(a1);
        B b1 = new B();
        System.out.println(b1);
        A a2 = b1;
        System.out.println(a2);
        B b2 = a2;
        System.out.println(b2);
    }
}
```

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

4. [30pts]

In this exercise, we consider a rectangular city, with streets running horizontally and avenues running vertically:

```
           ...   ...   ...  ... ...
...      |Av -2|Av -1|Av 0|Av 1|Av 2| ...
St  -2   |_____|__X__|____|____|____| ...
St  -1   |_____|_____|____|____|____| ...
St   0   |____ |_____|_Y__|____|____| ...
St   1   |_____|_____|____|____|____| ...
St   2   |_____|_____|____|_Z__|____| ...
...       ...    ...   ...  ...  ...
```

The city, which is similar to the ones you've seen in classes dealing with Becker's robots, extends indefinitely in all directions. Locations in this city are specified by a street number and an avenue number. For example, the location marked `X` above is at location (-2, -1), the one marked `Y` at (0, 0) and the one marked `Z` at (2, 1).

In this city, we can place various type of *vehicles* at given locations. We will only consider two types: cars and trucks. All vehicles have a name and can move north, east, south and west, but cars can also move north-west, north-east, south-east, and south-west. Trucks, on the other hand, can carry loads.

The following classes represent these vehicles. First, there is the base class `Vehicle`, which encapsulates the common functionality (basic construction with a name and an initial location, movement in the main cardinal directions, and string representation). Then, there are the derived classes `Car`, which adds movement in the intermediate directions, and `Truck`, which adds a new member variable, `load` and changes the string representation.

In this exercise you are asked to implement these three classes. You will do this by "filling in" the data and method definitions. You don't need to do any error checking or write comments, unless specifically asked to do so.

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 `Vehicle`, should appear as

```
+-------------- A --------------+
|  protected String name;        |
|  protected int street, avenue;  |
+--------------------------------+
```

and not squeezed on the page with the exam subjects!

Important: failure to clearly mark the blocks will result in deduction of points!

Remarks:

- Since block A has been filled in for you above, there are fifteen blocks left for you to fill in for this exercise: B, C, D1, D2, D3, D4, E, F, G1, G2, G3, G4, H, I, J. The related blocks D1, D2, D3, D4, and G1, G2, G3, G4, are 1 point each, as is H. Blocks B, C, and E are 3pts each, and blocks F, I, and J are 4pts each.
- Just to make sure, this is the table of the cardinal directions:

```
...        ...    ...    ...   ...
...       |Av_-1|Av_0_|Av_1|
St  -1    |_NW__|__N__|_NE_| ...
St   0    |__W_ |__X__|_E__| ...
St   1    |_SW__|__S__|_SE_| ...
...        ...    ...    ...   ...
```

  If the vehicle is at location `(0, 0)` (marked with `X`), then a move `N` takes it to `(0, -1)`, a move `SE` takes it to `(1, 1)`, etc.

```
public class Vehicle {
    // Block A
    // protected member variables:
    //     a string to hold the name of the vehicle
    //     integers for the street and avenue

    // Constructor
    public Vehicle(String name, int street, int avenue) {
        // Block B
        // Initialize the member variables to the given values
    }
    private void move(int dx, int dy) {
        // Block C
        // Change the member variables representing the
        // location of the vehicle with the given increments
        // (that is, add dx to the street coordinate and dy to the
        // avenue coordinate)
    }

    public void goN() {
        // Block D1
        // Use the private method move to change the location of
        // the vehicle one cell towards north.
    }

    public void goS() {
        // Block D2
        // Use the private method move to change the location of
        // the vehicle one cell towards south.
    }

    public void goW() {
        // Block D3
        // Use the private method move to change the location of
        // the vehicle one cell towards west.
    }

    public void goE() {
        // Block D4
        // Use the private method move to change the location of
        // the vehicle one cell towards west.
    }

    public String toString() {
        // Block E
        // return a string consisting of the name of the vehicle
        // and of its location.  If the vehicle is called
        // "My Vehicle" and it is on street -2 and avenue 3, printing
        // the string should result in
        //
        // Name: My Vehicle
        // Street: -2
        // Avenue: 3
    }
}


public class Car extends Vehicle {
    // no member variables

    // Constructor
    public Car(String name, int street, int avenue) {
        // Block F
        // Initialize the name and location of the car
        // to the given values
    }

    public void goNW() {
        // Block G1
        // Change the location of the car to the north-west.
    }

    public void goNE() {
        // Block G2
        // Change the location of the car to the north-west.
    }

    public void goSW() {
        // Block G3
        // Change the location of the car to the north-west.
    }

    public void goSE() {
        // Block G4
        // Change the location of the car to the north-west.
    }
}


public class Truck extends Vehicle {
    // Block H
    // Private member variable, an integer representing the current load
    // of the truck
    // Constructor
    public Truck(String name, int street, int avenue, int load) {
        // Block I
        // initialize the name, location, and load of the truck to the
        // given values
    }

    public String toString() {
        // Block J
        // return a string containing, besides the information
        // from the vehicle class, the load of the truck.  For example,
        // if the truck is called "My Truck", and is on street 5 and
        // avenue -2, with a load of 10000, printing the string should
        // result in
        //
        // Name: My Truck
        // Street: 5
        // Avenue: -2
        // Load: 10000
        //
    }
}
```

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) Write a main method in which you

   - [2pts] create a `Car` at location `(-2, 3)`,
   - [2pts] create a `Truck` at location `(2, 5)`,
   - [4pts] move the car and the truck to location `(6, 1)`

   In the last point, make sure you use the appropriate `go` functions, so that each vehicle makes the smallest amount of moves. Assuming each `go` move takes one step, irrespective of direction, who gets to `(6, 1)` faster (i.e., in the smaller number of steps)?([2pts])

   (b) Consider this main method:

```
public static void main(String[] args) {
    Vehicle v = new Vehicle("My Vehicle", 2, -3);
    System.out.println(v);
    v.move(1, 1);
    System.out.println("Street: " + v.street);

}
```

   Assume that the three classes above are in the same package. What lines need to be commented out to get the program to compile if the method is inserted

   i. in class `Vehicle`?
   ii. in class `Car`?
   iii. in a class in the same package as `Vehicle`, which is not a subclass of `Vehicle`?
   iv. in a subclass of `Vehicle` in a different package than `Vehicle`?
   v. in a class in a different package than `Vehicle`, which is not a subclass of `Vehicle`?

   Each correct answer brings 2pts.

6. [20pts] (Adapted from *CodingBat*)

   Write a method with the signature:

   ```
   int[] evenRun(int[] nums)
   ```

   For each even number in the given array, change all the values following it to be that even number, until encountering another even number. Examples:

   ```
   evenRun({7, 5, 8, 1})    ==>  {7, 5, 8, 8}

   evenRun({6, 1, 4, 3, 7}  ==>  {6, 6, 4, 4, 4}

   evenRun({8, 4, 6, 0})    ==>  {8, 4, 6, 0}

   evenRun({1, 5, 3})       ==>  {1, 5, 3}
   ```

   In the first example, we first find an even number, 8, at position 2 (the third number). We then find an odd number, and replace it with 8.

   In the second example, the first even number is 6, at position zero. The next number after that is odd, so we replace it with 6. At position 2 we have again an even number, 4, therefore there is no replacement. Then comes an odd number, 3, which gets replaced by the most recent even number, so by 4. The next (and last) number is again odd, 7, so it also gets replaced by 4.

   In the third example, all numbers are even, so there are no replacements.

   In the last example, we never find an even number, so no replacements are made either.

   Make sure to check the corner cases, e.g., when the array `nums` is empty!

   Remark: To check if a number is even, use the modulus operator (`%`). We have that `n % m` is the remainder of `n` divided by `m`. Thus, `n % 2` is zero if the `n` is even.