Exam
DIT948: Programming

Date:	2015-10-28
Time	8:30–12:30
Place	Lindholmen
Teacher	Musard Balliu
Examiner	Ivica Crnkovic
Questions	0762416640 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 are allowed
Reviewing:	2015-11-18 and 19, 13:00–15:00, Lindholmen Second floor room in Patricia (Note: subject to change, changes will be posted on the course website).

Please observe carefully 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 following Java program:

```
import static dit948.SimpleIO.*;

public class Swap {
        private int a;
        static int aGlobal = 1;

        public Swap(){}

        public Swap(int a){
                this.a = a;
        }

        static void swapInts(int a, int b){
                int tmp;
                tmp = a;
                a = b;
                b = tmp;
                aGlobal = a;
                println("Value of a in swapInts is "+a);
                println("Value of aGlobal in swapInts is "+aGlobal);
        }

        static void swapObjects(Swap x, Swap  y){
                int tmp;
                tmp = x.a;
                x.a = y.a;
                y.a = tmp;
                println("Value of o1.a in swapObjects is "+x.a);
        }

        public static void main(String[] args) {
                int a = 77;
                int b = 55;
                Swap o1 = new Swap(1);
                Swap o2 = new Swap();
                println("Initial value of a is "+a);
                println("Initial value of aGlobal is "+aGlobal);
                swapInts(a,b);
                println("Final value of a is "+a);
```

```
                println("Final value of aGlobal is "+aGlobal);
                println("Initial value of o1.a is: "+o1.a);
                swapObjects(o1,o2);
                println("Final value of o1.a "+o1.a);
                println("Final value of o2.a "+o2.a);
        }
}
```

What will be printed to the console (standard output) when running the program Swap?

2. [10pts] Your boss wrote a program that was supposed to compute the sum of all odd numbers and the sum of all even numbers up to (and including) a non-negative integer n. Recall that 0 is even and if n is 0, the result should be 0 in both cases. Otherwise, if for instance n is 5, then the sum of all odd numbers up to n is $1 + 3 + 5 = 9$ and the sum of all even numbers up to n is $0 + 2 + 4 = 6$.

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

```
public static int oddEven(int n){
        int sumEven=0;
        int sumOdd=1;
        int[] a = new int[n];

        for (int i=0; i<n; i++) {
                a[i] = i+1;
         }

        for (int i=0; i<n; i++) {
                if(i%2==0){ sumEven+=i; }
                else { sumOdd=a[i]; }
        }
        System.out.println("The sum of even numbers is "+ sumEven);
        System.out.println("The sum of odd numbers is "+ sumOdd);
}
```

3. [10pts] Consider the class:

```
package countries;

public class Country {

    private  String name;
    private  int population;
    boolean isEuropean;
    public double area;
    protected Country[] neighbors;

    protected boolean inEurope() { return this.isEuropean; }


    private void updatePopulation(int newBorns) {
        this.population += newBorns;
    }

    public String toString() {
        String str ="";
        for (int i=0; i<this.neighbors.length; i++){
           str += neighbors[i].name+"\n";
        }
        return str;
   }

   Countries[] getNeighbors() { return this.neighbors; }

   String getName() { return this.name; }
}
```

Class `TestCountry` is a subclass of `Country`:

```
1 package countries;
2
3 public class TestCountry extends Country {
4     public void run() {
5         System.out.println(name);
6         System.out.println(population);
7         System.out.println(isEuropean);
8         System.out.println(inEurope());
9         System.out.println(area);
```

```
10          System.out.println(toString());
11          updatePopulation(100);
12          System.out.println(getNeighbors());
13          System.out.println(getName());
14      }
15 }
```

Which of the lines 5–13 will cause a compilation error and why? How does the answer change if `TestCountry` is in a different package from `Country` (for example, if we replace the first line with `import countries.*;`)?

4. [30pts]

   In this exercise, you will implement several classes to represent all natural numbers consisting of at most 3 digits (all numbers from 0 to 999).

   You will do this by "filling in" the data and method definitions. You don't need to do any error checking or write comments.

   Please 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 variable of the class `Number`, should appear as

```
+----------A-----------+
|  private int n = 0;   |
+-----------------------+
```

   and not squeezed on the page with the exam subjects!

   Remarks:

   - Since block A has been filled in for you above, there are seventeen blocks left for you to fill in for this exercise. Blocks B, D, F, H J, N and O are worth one point each, blocks C, E, I, K, M, P and R are worth two points each, and blocks G, L and Q are worth 3 points each.

   End of remarks.

```
// Class representing 1-digit natural numbers (from 0 to 9)
```

```
public class Number {
  // Block A:
  // Declare a private instance variable:
  // n, 1-digit integer initialized to 0

        public Number(){
          this.n =0;
        }

        public Number(int n){
          //Block B:
          // Check that n is a 1-digit number,
          // if not, terminate using "System.exit(0)"
          // initialize instance variable to n
        }

        public Number (Number n){
         //Block C:
         // Initialize instance variable
        }

        public int getNumber(){
         //Block D:
         // Return the 1-digit number
        }

        public boolean equals(Number n){
         //Block E
         // Return true if this number is equal to n
         }

        public boolean compare(Number n){
          //Block F
          //return true if this number is greater than n
        }

        public String  toString(){
         // Block G
         // return a string consisting of this number
        }
```

```
// Class representing 2-digit numbers (from 00 to 99)
public class TwoDigitNumber extends Number {
         //Block H:
         // Declare two private instance variables
         // n1, Number representing first digit
         // n2, Number representing second digit

        public TwoDigitNumber(){
         //Block I
         // Initialize all digits to 0
         }

         public TwoDigitNumber(Number n1, Number n2){
         //Block J
         // Initialize all digits to n1 and n2
         }

         public boolean equals(TwoDigitNumber n){
         //Block K
         // Return true if this TwoDigitNumber number
         // is equal to n
         }

         public boolean compare(TwoDigitNumber n){
          //Block L
          //return true if this number is greater than n
          // Example: 21 > 11 and 10 > 08
          }

          public String toString(){
          //Block M
          // return a string consisting of this
          // TwoDigitNumber, for example 11 or 04.
          }

// Class representing 3-digit numbers (from 000 to 999)
public class ThreeDigitNumber extends TwoDigitNumber{
         //Block N:
         // Declare two private instance variables
         // n1, Number representing first digit
         // n2, TwoDigitNumber representing next 2-digits
```

```
        public ThreeDigitNumber(Number n1, TwoDigitNumber n2){
        //Block O:
        // Initialize all digits to n1 and n2
        }

        public boolean equals(ThreeDigitNumbers n){
        //Block P:
        // Return true if this ThreeDigitNumber number
        // is equal to n
        }

        public boolean compare(ThreeDigitNumber n){
        //Block Q
        //return true if this number is greater than n
        }

         public String toString(){
         //Block R
         // return a string consisting of this
         // ThreeDigitNumber, for example 111 or 003.
         }
}
```

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).

   Use the `dit948.Random` method `randomInt` to generate the random numbers needed in this exercise.

```
/**
 *  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);
```

   Assume you are given two methods with the signatures

```
public static Number getFirstDigit(int n);
public static Number getSecondDigit(int n);
```

that return a Number corresponding to the first and the second digit of a 2-digit natural number n, respectively. For example, if `n=75`, then `getFirstDigit(n)` returns the Number object corresponding to number `7` and `getSecondDigit(n)` returns the Number object corresponding to number `5`.

(a) Implement a method with the signature

```
public static Number[] genNums()
```

that returns an array of 100 random numbers of either 1-digit or 2-digit, namely from 0 to 99. You don't need to generate 3-digit numbers.

(b) Implement a method with the signature

```
public static String printMax(ThreeDigitNumbers[] nums)
```

that returns the string representation of the maximum number from an array of ThreeDigitNumber numbers.

Each method is worth 10 points.

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

Write a subroutine takes a string as argument and returns the sum of the digits 0-9 that appear in the string, ignoring all other characters. The subroutine returns 0 if there are no digits in the string. Note that Character.isDigit(c) tests if a character c is one of the characters '0', '1', .. '9' and Integer.parseInt(string) converts a string to an integer. Remember to check the corner cases! The subroutine should always return a value and not crash.

Examples:

```
sumDigits("aa1bc2d3") ==> 6
sumDigits("aa11b33") ==> 8
sumDigits("Chocolate") ==> 0
```