Programming – Final Exam January 2011 Leganés	 Universidad Carlos III de Madrid
Grado en Ingeniería Informática	

READ CAREFULLY THESE INSTRUCTIONS BEFORE STARTING THE EXAM:

- **Fill in all the pages with a pen (personal data and answers)**
- **Do not use a pencil or a red pen**
- **Do not forget your NIA and your actual group**
- **The duration of this exam is 3 hours**
- **Books and notes are allowed. Laptops, mobile phones, or any other electronic device are not allowed**

DO NOT CONTINUE WITH THE NEXT PAGE
until indicated

Surname	*Name*	
Signature	*NIA*	*Group*

Question 1 (1 Point)**.** The `main` method below prints on the screen `12.0`. Explain which method is being actually called –and why the remaining methods do not match the implementation of the `main`.

```
public static void main(String[] args) {
      int a=8;
      boolean b=false;
      double c=0;

      c = method1(a,b);
      System.out.println(c);

}
```

a)
```
public static void method1(int a, boolean b) {
      if (b == false)
            System.out.println(a+4);

}
```

b)
```
public static double method1(boolean b, int a) {
      double c;

      if (b == false){
            a = a+4;
      }
      c = a;

      return c;

}
```

c)
```
public static double method1(int x, boolean y){
      double w;

      if (y == false) {
            x = x+4;
      }
      w = x;

      return w;

}
```

d)
```
public static double method1(){
      return 12.0;

}
```

Question 2 (1 Point). Explain the output (i.e., what is printed on the screen) resulting from the execution of the `main` method of `Question2` class.

```
public class Question2 {

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

		System.out.println(++a);
		apply(a);
		System.out.println(a++);
	}

	public static void apply(int a) {
		a = a + 8;
	}

}
```

Problem 1 (2 Points). Create a class named `Problem1` and two static methods according to the following specification:

- (1 Point) `convert` method:
  - Parameters:
    - Receives an array of integers (named `list`)
  - Action:
    - Creates a new array with same size as `list`. The new array has the same values as `list`, but odd numbers are transformed into even numbers. (To do so, the method must traverse the array and add 1 if an odd number is found.) Each time an odd number is transformed, the method must print on the screen "Odd number found: *X*", being *X* the odd number
  - Return:
    - Returns the new array

  NOTE: The method must be able to deal with arrays of any size

- (1 Point) `main` method
  - Initializes an 10-element array with random integer values in {0, 99}
  - Calls the `convert` method
  - Prints the random and the converted array

EXAMPLE:

```
Odd number found: 65
Odd number found: 17
Odd number found: 45
Odd number found: 15

Initial array
65 16 17 86 80 45 88 15 50 52
Converted array
66 16 18 86 80 46 88 16 50 52
```

Problem 2 (2 Points). Create a class named `Problem2` and two static methods according to the following specification:

- (1.25 Points) `printPrimeNumbers` method:
  - Parameters:
    - Receives an integer value (named `n`)
  - Action:
    - Prints on the screen the prime numbers in the interval {2, `n`}
  - Return:
    - Nothing

  NOTE: A prime number is divisible only by 1 and itself.

- (0,75 Point) `main` method
  - Reads a number from the keyboard
  - Calls the `printPrimeNumbers` method
  - Repeats the operation until the user introduces a negative value

Problem 3 (2 Points). Given the code and the results below, implement the following static methods:

- (0,5 Points) `hasNegativeValues`:
  - Parameters:
    - A two-dimension array of integer numbers (named `m`). Notice that `m` may be an irregular array
  - Action:
    - Find outs if `m` has negative values
  - Return:
    - *True* if `m` has negative values; *false* otherwise

- (1 Point) `addDiagonal`:
  - Parameters:
    - A two-dimension array of integer numbers (named `m`)
  - Action:
    - If `m` is regular and square, the method adds the values of the elements at its main diagonal
  - Return:
    - Result of the sum; 0 if `m` is not a regular and square matrix

- (0,5 Points) `addLowerHalf`:
  - Parameters:
    - A two-dimension array of integer numbers (named `m`)
  - Action:
    - Adds the values of the elements at the lower half of `m`. If `m` rows number is odd, the medium row have to be considered
  - Return:
    - Result of the sum

NOTE: A regular matrix is a matrix such as all its rows have the same number of columns. A square matrix is a matrix that has the same number of rows and columns

SAMPLE `main` METHOD:

```
public static void main(String[] args) {

      int [][] testMatrix1 = new int[][] {
                  { 2,  3,   9,  4},
                  { 4,  1,   3,  5},
                  { 1,  9,   9,  9},
                  {-7,  8,   0,  1}
            };  // square and regular matrix

      int [][] testMatrix2 = new int[][] {
                  { 3,  3,  8,  2},
                  { 1,  1,  4,  4},
                  { 1,  9,  9,  9, 2, 2},
                  { 7,  8,  0,  2}
            };    // square irregular matrix

      boolean tmp;

      tmp = hasNegativeValues(testMatrix1);
      System.out.println("1st result: " + tmp);
      tmp = hasNegativeValues(testMatrix2);
      System.out.println("2nd result: " + tmp);

      int sum;
      sum = addDiagonal(testMatrix1);
      System.out.println("3rd result: " + sum);
      sum = addDiagonal(testMatrix2);
      System.out.println("4th result: " + sum);

      sum = addLowerHalf(testMatrix1);
      System.out.println("5th result: " + sum);
      sum = addLowerHalf(testMatrix2);
      System.out.println("6th result: " + sum);

}
```

RESULT:

```
1st result: true
2nd result: false
3rd result: 13
4th result: 0
5th result: 30
6th result: 49
```

Problem 4 (2 Points).

Create a public class named `Seat` according to the specification below:

- (0.1 Points) Define the following public attributes:
  - `id` (String) [seat identifier]
  - `taken` (boolean) [is the seat taken?]

- Create the following public non-static methods:

  - (0.4 Points) `setId`. Calculates and assign an identifier to the seat according to the *row* and *column* values passed as parameters. The method does not return any value.

    For instance, if the method is called as `setId(2, 4)` [*row 2*, *column 4*], the value `"2D"` must be assigned to the `id` attribute. The first valid row and column value is 1.

    NOTE: Reuse the following code snippet to calculate the letter corresponding to a *column* integer value:

    ```
    String letter = String.valueOf((char)(column+ 'A' - 1));
    ```

  - (0.1 Points) `getId`. Returns a String corresponding to the identifier of the seat. (Assume that the value has been previously assigned with `setId`.) The method has no parameters.

  - (0.2 Points) `setTaken`, `isTaken`. Assign and retrieve the `taken` attribute, respectively.

Create a public class named `TestSeat` according to the specification below:

- Create a `main` method

- (0.4 Points) The program must declare and allocate memory for a two-dimension array of *r* x *c* `Seat` objects (named `wagon`). *r*, *c* values must be read from the keyboard. (This array represent a train wagon with *r* seat rows, *c* seats per row.)

- (0.2 Points) Initialize the objects of the `wagon` array. Proper values to the `id` and `taken` attributes of each seat of the wagon must be assigned (use the `setId` and `setTaken` methods). Initially, all the seats must be free.

- (0.1 Points) Mark all the seats of the last row of the wagon as taken (use the `setTaken` method).

- (0.4 Points) Create a static method named `countFreeSeats` with the following specification:

  - Parameters:
    - A two-dimension array of `Seat` representing a train wagon (named `w`)
  - Action:
    - Calculates the number of free seats in the wagon
  - Return:
    - Number of free seats

- (0.1 Points) Extend the main method to print on the screen the number of free seats of wagon (use the `countFreeSeats` method.)

Programming – Grado en Ingeniería Informática

Authors

Of the English version:
Juan Gómez Romero

Based on the work by:
Ángel García Olaya
Manuel Pereira González
Silvia de Castro García
Gustavo Fernández-Baillo Cañas
Daniel Pérez Pinillos
Javier Ortiz Laguna
Álvaro Torralba Arias de Reyna

Universidad
Carlos III de Madrid