Programming – Final Exam July 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 (0.5 Points) Explain the output (i.e., what is printed on the screen) resulting from the execution of the `main` method of the class below:

```
public class Question1 {

	public static void main(String[] args) {
		int a = 3;
		int b = 2;

		b *= a++;
		System.out.println(a);
		System.out.println(b);
	}
}
```

The output is:

4
6

a is incremented with the post-increment operator. Therefore, the expression can be transformed into:

*b = b * a;*
a = a + 1;

Question 2 (1 Point) Are the following questions right? Explain your answer.

1. (0.25 Points) Java methods must include at least one `return` instruction.
   False. void method may not include a return instruction

2. (0.25 Points) The following code snippet results in an infinite loop without regard of the instructions included inside the block.

```
while(true) {
      instructions;
}
```

   False. A break can be included in the block, thus stopping the execution of the while loop

3. (0.25 Points) Values can be assigned to an array in the same sentence in which the array is created.
   True. For example, int [] a = new int[]{1, 2, 3, 4, 5}; creates, allocates and initializes an array

4. (0.25 Points) *Bubblesort* is one of the most efficient sorting algorithms.
   False. Bubblesort is one of the less efficient sorting algorithms

Question 3 (1 Point) Given the code snippet below, indicate the result of the following calls to the method `method`:

```
public static int method(int [] values, char character) {
      if(values.length > 5) {
            System.out.println("Array is too long.");
      } else {
            for(int i=0; i< values.length-1; i++) {
                  if(i != 0) {
                        System.out.print(",");
                  }
                  System.out.print(values[i]);
            }
            return -1;
      }
      System.out.println("Caracter: " + character);

      return 0;

}
```

Notice the termination condition of the for loop: values.length-1. Accordingly, the loop does not access to the last element of the values array.

a) `method(new int[] {1, 2, 3, 4}, 'a');`

b) `method(new int [] {1, 2, 3, 4, 5, 6}, 'b');`

c) `method(new int [] {1}, 'c')`

d) `System.out.println(method(new int[]{1, 2, 2}, 'e'));`

Problem 1 (2.5 Points) Create a class named `Problem1` with the following static methods:

- (0.5 Points) Method `percent`:
  - Parameters: 1-dimension array of `double` named `list`
  - Action: Counts how many values of `list` are larger or equal that the first element
  - Returns: `double` value resulting from dividing the number of values which are larger or equal that the first one by the total length of the array

- (0.5 Points) Method `largers`:
  - Parameters: 1-dimension array of `double` named `list`. `double` value `v`
  - Action: Finds the elements of `list` which are larger or equal to `v`
  - Returns: 1-dimension array of double values including the values of `list` larger than `v`

  EXAMPLE:
  list = {3.1, 5.2, 6.5, 2.3, 1.0}, v = 2.1
  result = {3.1, 5.2, 6.5, 2.3}

- (1 Point) Method `findLocalMaximums`:
  - Parameter: 1-dimension array of `double` named `list`
  - Action: Finds the local maximum values of `list`. A "local maximum" is a value larger than both the previous value and the next one. (The first element of the array must be compared only with the second element; the last value of the array must be compared only with the last but one element)
  - Returns: 1-dimension array of `double` with the local maximum values

  EXAMPLE:
  list = {3.1, 5.2, 6.5, 1.0, 2.3}
  result = {6.5, 2.3}

- (0.5 points) Method `main`:
  - Action: Declares and allocates a 50-element array of `double` (`z`). The elements are assigned a random value in [0, 100). Prints the array `z`. Calls the method `findLocalMaximums` using `z` as parameter; prints the resulting array (`w`). Call the method `largers` using `w` as parameter (the other parameter is any number directly specified by the student –it is not necessary to read it from the keyboard); prints the resulting array (`y`). The array `y` is used to call the `percent` method; prints the resulting value (`r`)

Problem 2 (2.5 Points) Create a class named `Card` according to the following specification:

- (0.4 Points) Public static attributes:
  - `SUITS`: array of `String` with the values: `"Spades"` (♠), `"Hearts"` (♥), `"Diamonds"` (♦),`"Clubs"` (♣)
  - `NAMES`: array of `String` with values: `"ace"`, `"two"`, `"three"`, `"four"`, `"five"`, `"six"`, `"seven"`, `"eight"`, `"nine"`, `"ten"`, `"knave"`, `"queen"`, `"king"`
- (0.2 Points) Private attributes:
  - `suit`: `String` (suit of this card)
  - `name`: `String` (name of this card)
- (0.6 Points) Method `isIn`:
  - Parameters: One-dimension `String` array (`a`), `String` (`v`)
  - Action: Checks if `v` is equal to any of the elements in the array `a`
  - Returns: `v` if `v` is found in the array `a`; otherwise, the method returns the first element of `a`
- (0.4 Points) Method `setSuit`:
  - Parameter: `String s` representing the *suit* value to assign
  - Action: Checks if `s` is a valid suit; to do so, it calls the `isIn` method with `SUITS` and `s` as parameters. If `s` is in `SUITS`, it assigns `s` to the `suit` attribute
  - Returns: Nothing
- (0.4 Points) Method `setName`:
  - Parameter: `String n` representing the *name* value to assign
  - Action: Checks if `n` is a valid name; to do so, it calls the `isIn` method with `NAMES` and `n` as parameters. If `n` is in `NAMES`, it assigns `n` to the `name` attribute
  - Returns: Nothing
- (0.3 Points) Method `getSuit`:
  - Parameter: None
  - Action-Returns: Returns the value of the attribute `suit`
- (0.2 Points) Method `getName`:
  - Parameter: None
  - Action-Returns: Returns the value of the attribute `name`

Problem 3 (2 Points) Create a class named `Deck` according to the following specification:

- (0.2 Points) Private attributes:
  - `cards`: 52-element array of `Card` objects

- (0.3 Points) Method `obtainRandomSuit`:
  - Parameters: None
  - Action: Picks a random *suit* value `s` (from the allowed values)
  - Returns: `String` value `s`

- (0.3 Points) Method `obtainRandomName`:
  - Parameters: None
  - Action: Picks a random *name* value `n` (from the allowed values)
  - Returns: `String` value `n`

- (0.5 Points) Method `createCard`:
  - Parameters: *name* (`String`) and *suit* (`String`) values
  - Action: Creates a new `Card` object and sets the specified *name* and *suit*
  - Returns: The new `Card` object

- (0.5 Points) Method `initializeDeck`:
  - Parameters: None
  - Action: Allocates and initializes the `Card` array `cards` with 52 random cards (use the `obtainRandomSuit` and `obtainRandomName` methods) (NOTE: The new deck does have to include all the possible cards of a complete deck)
  - Returns: Nothing

- (0.2 Points) Method `printDeck`:
  - Parameters: None
  - Action: Prints the information of the cards in the `cards` array; e.g.: `ace of Spades, three of Hearts, king of Hearts`, etc.
  - Returns: Nothing

Problem 4 (0.5 Points) Create a class named `TestDeck` with a `main` method that creates a `Deck` object `d`, initializes `d`, and prints the information of the cards. (NOTE: Use the methods developed in Problem 3.)

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