IIP First Partial - ETSInf
November 10th, 2014. Time: 1 hour and 30 minutes.

It is wanted to implement an application that represents a block game, with blocks of different colors and dimensions, that can be stacked up in towers. Each block has associated a **color** (blue or red) and a **dimension** (integer number in range 1 to 50, both included).

The game rules say that:

- Blocks stacked up in a tower must follow alternate colors (on a blue block only red blocks can be stacked, and vice versa)
- On a block of dimension x only a block of dimension y, with $y \leq x$, can be stacked up (i.e., the tower becomes narrower towards the top, and wider towards the base)
- A block can be a **wildcard**; in that case, it can be stacked on any other block independently of the block color, but accomplishing the dimension restriction

1. [6 points] You must implement the `Block` class, specifically:
   a) (0.5 points) Declare public constant class attributes for representing the two possible colors as integers: `BLUE` and `RED`, with values 0 and 1 respectively. These constants must be employed when required (in both the `Block` and the `BlockTower` classes)
   b) (0.5 points) Declare private object attributes for `color` (`int`), `dimension` (`int`), and `wildcard` (`boolean`)
   c) (1.5 points) Implement two constructors:
      - A general constructor with as many parameters as needed to initialise all object attributes
      - A default constructor that creates a blue `Block`, which is not a wildcard and whose dimension is given by a random number in the range [1,50]
   d) (0.5 points) Implement the consultor and modifier methods for the `dimension` attribute
   e) (1 point) Override the `equals` method of the `Object` class to check if two blocks are equal; one block is equal to other is both them are blocks and all their attributes coincide
   f) (1 point) Override the `toString` method of the `Object` class to show it result in a format similar to the following examples: "`(Color: red, dimension: 22 and it IS wildcard)`", "`(Color: blue, dimension: 15 and it IS NOT wildcard)`"
   g) (1 point) Implement a method `canBeOn(Block b)` such that returns true or false depending on the current `Block` can be on the `Block b` that receives as parameter, depending on the rules stated above. For example, given `Block` variables `a` and `b`, `a.canBeOn(b)` is true only when dimension of `a` is lower or equal than dimension of `b` and `a` is a wildcard or has a color different from that of `b`

Solución:

```
public class Block {
    private int color;
    private int dimension;
    private boolean wildcard;

    public static final int BLUE = 0;
    public static final int RED  = 1;
```

```
    public Block() {
       this.color = BLUE;
       this.dimension = (int) (1 + Math.random() * 50);
       this.wildcard = false;
    }

    public Block(int color, int dimension, boolean wildcard) {
       this.color = color;
       this.dimension = dimension;
       this.wildcard = wildcard;
    }

    public int getDimension() { return dimension; }

    public void setDimension(int dim) {
       dimension = dim;
    }

    public boolean canBeOn(Block b) {
       return this.dimension <= b.dimension
              && (this.wildcard || this.color != b.color);
    }

    public boolean equals(Object o) {
       return o instanceof Block
              && this.color == ((Block) o).color
              && this.dimension == ((Block) o).dimension
              && this.wildcard == ((Block) o).wildcard;
    }

    public String toString() {
       String col = "blue";
       if (this.color == RED) col = "red";
       String wc = "IS NOT";
       if (this.wildcard) wc = "IS";
       return "(Color: "+ col + ", dimension: " + dimension + " and "
              + wc + " wildcard)";
    }
}
```

2. 4 points Implement a class `BlockTower` whose aim is to test small towers with a few blocks. For that, you must implement that class with a method `validDimension` that:

   a) (1.25 points) From a given dimension, limits its value to the corresponding limits of the interval [1,50], i.e., when the given value is lower than 1, 1 is returned, and when the value is higher than 50, 50 is returned. Otherwise, the returned value is not modified.

   and a `main` method that makes the following actions:

   a) (0.25 points) Create a `Block` object `b1` with the default constructor

   b) (0.5 points) Create a `Block` object `b2` with the general constructor, with color blue, dimension 30, and wildcard

c) (1 point) Create a `Block` object `b3` with the general constructor and with data (previously asked to the user by employing the standard input) for color and dimension (it will not be a wildcard); color must be asked as a `String` (with values "red" or "blue", when a different value is inputted color will be red); dimension must be an integer in [1,50] and, in order to achieve that, the previously defined method `validDimension` must be employed to obtain the integer in that range.

d) (0.25 points) Show on the screen the three created objects

e) (0.75 points) Determine and show on the screen whether the tower formed by stacking up block `b3` on `b2`, and this last on `b1`, is a valid tower or not

Import the classes that you consider you need, and employ the constants defined in class `Block` when required.

Solución:

```
import java.util.Scanner;
public class BlockTower {

    public static int validDimension(int dimension) {
       if (dimension < 1) dimension = 1;
       else if (dimension > 50) dimension = 50;
       return dimension;
    }

    public static void main(String[] args) {
       Scanner kbd = new Scanner(System.in);
       Block b1 = new Block();
       Block b2 = new Block(Block.BLUE, 30, true);

       System.out.print("Input color for block 3 (blue/red): ");
       String color = kbd.next().toLowerCase();
       int codColor = Block.RED;
       if (color.equals("blue")) codColor = Block.BLUE;

       System.out.print("Input dimension for block 3 [1,50]");
       int dimension = validDimension(kbd.nextInt());

       Block b3 = new Block(codColor, dimension, false);

       System.out.println("Block 1: " + b1);
       System.out.println("Block 2: " + b2);
       System.out.println("Block 3: " + b3);

       System.out.print("\nTower formed with those three blocks");
       if (b3.canBeOn(b2) && b2.canBeOn(b1))
            System.out.println(" is valid");
       else System.out.println(" is NOT valid");
    }
}
```