dit UPM	ETSIT-UPM. PROGRAMACIÓN. CURSO 2011-2012. EXAMEN 2. APELLIDOS: NOMBRE: EMAIL:	

Pregunta 1. (2 puntos) Escriba un método para imprimir por pantalla los enteros del 1 al *n* (número pasado como parámetro) en filas de 10 números. El método lanza Exception si *n* es menor o igual que 0.	Use un bucle while para calcularlo. Por ejemplo, si *n* es 14, el método debe escribir: 1 2 3 4 5 6 7 8 9 10 11 12 13 14

```
void imprimePorDecenas (int n) throws Exception{
  if (n <=0) {
     throw new Exception();
  }
  int i = 1;
  while (i<=n) {
     System.out.print(i+" ");
     if (i%10 == 0) System.out.println();
     i++;
  }
}
```

Pregunta 2. (4 puntos). El código que identifica a una oficina bancaria se representa con una String que contiene 9 dígitos: los cuatro primeros indican el banco, los siguientes cuatro la oficina y el último para comprobar si los anteriores son correctos (digito de control). Escriba un método con un parámetro String de código bancario que devuelve true si es correcto y no lanza excepciones. Aplique este algoritmo:
- tomar la String compuesta por los 8 primeros dígitos y añadirle la String "00" por la izquierda,
- recorrer la nueva String multiplicando el valor de cada dígito por el valor correspondiente en un array *int[] valores = {1, 2, 4, 8, 5, 10, 9, 7, 3, 6}*, sumando los resultados
- se calcula 11 menos el resto de dividir el resultado **de la suma** anterior por 11
- si el resultado que se obtiene es 11, se cambia a 0; si el que se obtiene es 10, se cambia a 1
- y el resultado del método se obtiene comparando el valor obtenido con el valor del dígito de control.

El carácter en la posición i de una String se obtiene usando el método String.charAt(i). Para obtener una parte de una String puede usar el método String.subString (int primer, int ultimo) que devuelve otra String que empieza en la posición primero y acaba justo antes del último. Para calcular el valor entero a partir de un carácter puede usar Integer.parseInt(char) throws Exception.

```
boolean comprueba (String c) {
try {
  if (c.length() != 9)
    return false;
  String cod = "00" + c.substring (0, 8);
  int [] v = {1, 2, 4, 8, 5, 10, 9, 7, 3, 6};
  int s = 0;
  for (int i = 0; i < cod.length(); i ++) {
     s += v[i] * Integer.parseInt(""+cod.charAt(i));
  }
  s = 11 – ( s % 11 );
  if (s == 11)
    s = 0;
  if (s == 10)
    s = 1;
  return (s == Integer.parseInt(""+c.charAt(8)));
} catch (Exception e) {
  return false;
}
}
```

dit UPM	ETSIT-UPM. PROGRAMACIÓN. CURSO 2011-2012. EXAMEN 2. APELLIDOS: NOMBRE: EMAIL:	

Pregunta 3. (2 puntos). Complete la clase Bombilla, que representa una bombilla de un edificio con su identificador y la potencia que consume. Debe completar el constructor y los métodos encender y apagar. Las bombillas devuelven true o false si se han podido encender o apagar o no. Una bombilla no se enciende si la potencia total consumida con su encendido supera los 5000W que se establecen para todas las bombillas.

```
class Bombilla {
  private int identificador;
  private double potencia;
  public Bombilla (int id, double potencia) { … }
  public boolean encender() { … }
  public boolean apagar () { …}
}
```

```
class Bombilla {
  private static final double POTENCIA_MAXIMA= 5000.0;
  private static double potenciaActual = 0.0;
  private int identificador; // en realidad sobra
  private double potencia;
  private boolean encendido;

 public Bombilla (int id, double potencia) {
     this.identificador = id;
     this.potencia = potencia;
     encendido = false;
 }
 public boolean encender () {
     if (potencia + potenciaActual <= POTENCIA_MAXIMA) {
         potenciaActual += potencia;
         encendida = true;
         return true;
     }
     return false;
 }
 public boolean apagar() {
     if (encendida) {
          potenciaActual -=potencia;
          encendida = false;
          return true;
     }
     return false;
 }
} // NOTA: para la corrección se han dado por buenas soluciones que no incluyen el atributo encendido
```

Pregunta 4. (2 puntos) Dado el tipo enumerado Color que se muestra a continuación, defina un método getColor que tome como parámetro un entero y devuelva: el color *rojo* si el entero vale 0; el color *amarillo* si el entero vale 1; el color *verde* si el entero vale 2; y lance una excepción en cualquier otro caso con el mensaje de "Color no válido", *public enum Color {rojo, amarillo, verde};*

```
public Color getColor(int i) throws Exception {
        switch (i) {
                case 0: return Color.rojo;
                case 1: return Color.amarillo;
                case 2: return Color.verde;
                default:
                        throw new Exception ("Color no válido");
                }
        }
}
```