


Prueba Recuperación 2º Trimestre - Tiempo 2h	14 de Mayo de 2020 - 16:00h
Nombre y Apellidos	DNI/NIE: Firma:
1º Desarrollo de Aplicaciones Web (Vespertino) Módulo: Entornos de Desarrollo (Unidades 3 y 4)	 IES Alonso de Avellaneda (Alcalá de Henares)

No se permite ningún material de estudio o notas de clase. Para la parte práctica se utilizará una conexión internet sólo para el github y la correspondiente MV para los programas que se entreguen.

Se requerirán dos entregas: una parcial y otra definitiva a lo largo del examen.

Problemas (10 puntos)

Todos los ficheros necesarios están en el repositorio de la prueba.

1. Según el código facilitado (4p):

```
public int static Contador(int x, int y) {
    Scanner entrada = new Scanner(System.in);
    int num, c = 0;
    if ( x > 0 && y > 0) {
        System.out.println("Escribe un número");
        num = entrada.nextInt();
        if (num >= x && num <= y) {
            System.out.println("\tNúmero en el rango");
            c++;
        }
        else
            System.out.println("\tNúmero fuera de rango")
    }
    else
        c = -1;
    entrada.close();
    return c;
}
```

- Realiza el grafo de complejidad ciclomática completo escribiendo los nodos, flechas y condiciones (2p)
- Calcula la complejidad ciclomática de las tres formas indicando las fórmulas que se necesitan. (0.5p)
- Define el conjunto básico de caminos indicando los nodos de cada uno (0.5p)
- Elabora los casos de prueba para cada camino (1p)

2. Se tiene una función en la que se devuelve el texto "Par" si el número es un número entero par y se devuelve el texto "Impar" si el número fuese impar. Elabora la tabla con las clases de equivalencia válidas y no válidas poniéndoles un identificador. Luego, realiza la tabla con los casos de prueba, las clases de equivalencia, condiciones de entrada y resultados

esperados. (2p)

```
public String parImpar(int num){
    String cad="";
    if ( num % 2 == 0)
        cad = "Par";
    else
        cad = "Impar";
    return cad;
}
```

3. Crea una batería de pruebas parametrizadas y aserciones con valores límite que verifiquen la salida de dicho método mediante JUnit4. (2p)

```
public class Ejercicio3 {

    public static int sumaEnterosPositivos(int x, int y) {
        if ( x > 0 && y > 0) {
            return x + y;
        }
        else if (x < 0 || y < 0) {
            return -1;
        }
        else if ( y == 0 && x == 0 ) {
            return 0;
        }
        return -2;
    }
}
```

4. Crea un repositorio en tu sitio github llamado `xedr2` (donde x es nombre alumno) . Realiza los comandos necesarios con las salidas de los mismos.

a) Crea las carpetas y enlázalo a la siguiente carpeta previamente creada `~/repoexamen/xED2R`. Dentro de la anterior ubicación crea la carpeta **app1** y luego crea el fichero `Test.java` dentro de la misma que imprima un sencillo mensaje que sea `Rama principal`. Sube todos los cambios al remoto y muestra el log del git del repositorio local antes y después de subir los cambios. (1p)

b) Crea la rama llamada `secundaria` cámbiate a ella y modifica el fichero `Test.java` para que muestre otro mensaje que será `Rama secundaria`. Muestra en qué rama estás y muestra la salida del fichero java. Cámbiate a la rama main y vuelve a mostrar el fichero java. Fusiona después con la rama master y muestra la salida del fichero java. (1p)

Rúbrica: 100% si la respuesta está totalmente correcta y se ajusta a la especificación dada. En caso de cuestión incorrecta, ilegible, no clara, no precisa, no se ajusta, muy deficiente: 0%. Si la respuesta está desarrollada pero tiene deficiencias leves ante su especificación se podrá valorar hasta 50%.

No se puede extraer información de recursos de Internet, ni de material de clase y la prueba tiene carácter individual