


Integrador Java

Integrador Teórico y Practico de Java.

En este formulario van a encontrar 10 preguntas teóricas para responder y al final van a encontrar 3 problemas a resolver en Java, estos se van a resolver en Netbeans con un proyecto que se les va a entregar mediante un pdf y la idea es que ustedes suban la resolución a cada problema en las casillas que corresponda. Hacer primero la parte practica y después volver a responder el formulario. Esto después lo que corregirá el profesor y les enviará al mail el resultado final.

 ~~XXXXXXXXXX~~@gmail.com (no compartidos) [Cambiar de cuenta](#)



***Obligatorio**

Nombre y Apellido *

Tu respuesta

Correo electrónico *

El correo a donde te enviaremos la devolución del integrador

Tu respuesta

1- ¿Qué palabra se usa en Java para instanciar un objeto de una clase? *

10 puntos

- ☐ Ninguna de las anteriores
- ☒ new
- ☐ create
- ☐ Cualquiera de las anteriores



2- Las colecciones en Java son: *

10 puntos

- ☐ Un grupo de objetos
- ☒ Un grupo de clases
- ☐ Un grupo de interfaces
- ☐ Ninguna de las anteriores.

3- El modificador de acceso private, hace que los datos puedan ser accedidos por *

10 puntos

- ☐ Cualquier clase
- ☒ La clase donde se encuentran
- ☐ El metodo main
- ☐ Ninguna de las anteriores.

4- ¿Para qué sirve la palabra reservada final en Java? *

10 puntos

- ☐ Para indicar que un atributo solo puede tener valores numéricos
- ☒ Para indicar que una variable no puede cambiar su valor.
- ☐ Para indicar que esa clase hereda de otra clase.
- ☐ Ninguna de las anteriores



5- Las relaciones en Java son entre: *

10 puntos

- ☐ Interfaces
- ☐ Métodos
- ☒ Clases
- ☐ Ninguna las anteriores.

6- ¿Cuál es la principal característica de un Set? (Colecciones) *

10 puntos

- ☐ Puede contener objetos de diferentes clases.
- ☒ No se puede insertar dos veces el mismo objeto.
- ☐ Una vez que insertamos un objeto, no podemos eliminarlo.
- ☐ Tiene una capacidad predefinida.

7- Las excepciones se manejan con el bloque: *

10 puntos

- ☐ Finally
- ☐ Throws
- ☒ Try catch
- ☐ Throw

8- ¿Cuál es una característica de las interfaces? *

10 puntos

- ☐ No se pueden instanciar
- ☐ Solo estan compuestas de metodos
- ☒ Puede tener atributos, pero solo si estos son constantes.
- ☐ Todas las anteriores.



9- ¿Qué es una clase abstracta? *

10 puntos

- ☐ Una clase que no tiene atributos
- ☐ Una clase que no puede heredarse a otras clases.
- ☐ Una clase que no tiene constructores.
- ☒ Una clase que no se puede instanciar

10- ¿Para qué sirve la palabra reservada throws? *

10 puntos

- ☒ Para indicar que un método puede largar una excepción.
- ☐ Para largar una excepción desde dentro de un método.
- ☐ Para capturar excepciones que largue un método.
- ☐ Todas las anteriores

Función numeroCapicua(): La función recibirá un numero x y deberá determinar si es capicúa o no. Este deberá devolver verdadero(true) si es capicúa y falso(false) si no lo es. Además deberemos contemplar los siguientes escenarios: • Contemplar que el numero que llega puede ser negativo. • Contemplar que el numero que llega puede ser de un dígito, si es así debe devolver true. • Contemplar que el numero que llega puede ser null, si es así debe devolver false. *

33 puntos

Tu respuesta

```
public static boolean numCapicua(Integer numero) {
    Integer aux = numero, cifra = 0, inverso = 0;
    boolean verificar = false;
    if (aux == null) {
        verificar = false;
    } else {
        if (aux > 0 && aux < 10) { // Verifico que no sea un solo digito
            verificar = true;
        } else {
            while (aux != 0) { // si el numero es distinto de 0 recorro numero
                cifra = aux % 10; // saco el resto del numero
                inverso = (inverso * 10) + (cifra); // Voy dando vuelta el numero 1 a uno para ver si es capicua
                aux = aux / 10; // Voy dividiendo hasta que queda en cero para dejar de repetir el proceso
            }
            if (numero.equals(inverso)) { // Verifico si numero es igual a inverso que es el que fui dando vuelta
                // entonces es capicua si no no es capicua
                verificar = true;
            } else {
                verificar = false;
            }
        }
    }
    return verificar;
}
```

Función prisioneroDulce(): Estamos en caramelolandia, donde estan los peores ladrones de dulces. Una vez al mes, se sienta una n cantidad de presos en ronda, contemplando al preso que inicia la ronda, como el preso 0. A los presos se les da una m cantidad de caramelos, estos caramelos se repartirán de uno en uno a cada preso, contemplando que se comenzaran a repartir los caramelos desde el primer preso (inicio). Se repartirán los caramelos hasta que no queden más y el ultimo caramelo en repartirse estará podrido, determinar a que preso, según su posición en la ronda le tocara el caramelo podrido. *

Tu respuesta

Respondo a estas dos funciones al final con imagenes

Función mediasAmigas(): En un universo paralelo, donde los habitantes son medias, existe un crucero de medias donde se sube una lista de medias. Esta lista de tripulantes del crucero es una Collection de letras. Lo que se deberá hacer, es filtrar la lista de medias que se suben al crucero y retornar una lista que contenga los grupos de medias que si tenían pares. Esta lista final de medias pares se mostraran ordenadas de menor a mayor. A continuación un ejemplo: List de medias que llegan : A,B,A,B,C,A,F,Z,C,H. A,B y C tiene pares, mientras que F,Z y H no. Entonces la List que se debería retornar sería: A,B,C. *

Tu respuesta

Comentarios

Algún comentario que quieras hacernos acerca de este integrador

Tu respuesta

Submission ID (skip this field) *

⚠ DO NOT EDIT this field or your time will not be recorded.

Tu respuesta

Enviar

[Borrar formulario](#)

Nunca envíes contraseñas a través de Formularios de Google.



Este contenido no ha sido creado ni aprobado por Google. [Notificar uso inadecuado](#) - [Términos del Servicio](#) - [Política de Privacidad](#)

```

public static int prisioneroDulce(int inicio, int cantCaramelos, int cantDePresos) {
    int resultado = 0;
    for (int i = inicio ; inicio < cantDePresos; i++) { //instancio i con la variable inicio
        cantCaramelos--; //resto 1 caramelo por cada vuelta
        if(cantCaramelos == 0) { //si los caramelos son igual a cero
            resultado = i; //guardo el prisionero que recibio el podrido
            break;
        }
        if(i == cantDePresos) { //si la i es igual a la cantidad de preso reinicio bucle ya que caramelos no es == a 0
            i = 0;
        }
    }
    return resultado;
}

```

```

public static HashSet<String> mediasAmigas(ArrayList<String> listaMedias){
    int fin = listaMedias.size(); //tomo el tamaño de la lista para recorrerla
    HashSet<String> result = new HashSet();
    for (int i = 0; i < fin; i++) {
        int cant = Collections.frequency(listaMedias, listaMedias.get(i)); //Recorro la lista, con frequency toma los
        //datos que se repiten en la lista y el segundo parametro el dato que se repite, en este caso la letra que
        //adquiero con listamedias.get(i)
        if(cant >= 2) { //si la cantidad es mayor o igual a 2 hay un par tonces la guardo en uns hashset
            //las guardo en un hashSet para que no me repita letras
            result.add(listaMedias.get(i));
        }
    }

    return result; //retorno hasset y lo muestro por pantalla
}

```

