Esta hoja de ejercicios nos servirá para repasar las nociones más importantes del Tema 5.

1. ¿Cuál es el nombre en Java de la clase que define las excepciones, y de la que debe heredar cualquier clase que queramos usar para representar una excepción?

2. ¿Cuál es el nombre en Java de la clase que representa las excepciones que se producen al invocar un método de un objeto cuyo valor es "null"?

3. ¿Cuál es el nombre en Java de la clase que representa las excepciones que se producen al obtener un comportamiento anómalo en la entrada / salida de información?

4. Observa el siguiente fragmento de código:

```
String [] array_string = new String [25];
System.out.println (array_string [3].length());
```

¿Qué excepción se produciría en el mismo?

5. Observa el siguiente fragmento de código:

```
String aux = "hola";
int aux2 = Integer.parseInt (aux);
```

¿Qué sucedería al ejecutar el mismo?

6. Escribe un método auxiliar de nombre "caracterEn" en Java que realice la siguiente acción:

Recibe como parámetros una cadena (String) y un entero (int);

Si el entero está entre 0 y la longitud de la cadena (puedes hacer uso del método "length(): int" de la clase "String") devuelve el carácter en la posición correspondiente (puedes hacer uso del método "charAt(int)" de la clase "String").

En caso contrario, construye y lanza una excepción de tipo "Exception".

7. Construye un programa "main" en Java que realice las siguientes acciones:

- Crea un objeto de la clase "Scanner" y lo asocia con la entrada estándar (la consola de MSDOS);
- Lee un objeto de tipo "String" de la misma en un objeto "lect_teclado";
- Invoca al método "caracterEn" definido en el ejercicio anterior, con la "String" leída de la entrada estándar y el entero "7", mostrándolo por pantalla;
- Captura la posible excepción, mostrando por pantalla un mensaje: "Has intentado recuperar una posición de la cadena de caracteres que no existe;".

8. Define una clase "NumeroNegativoExcepcion" que herede de "Exception" y que contenga un constructor sin parámetros y un constructor que reciba como parámetro una "String", de tal modo que ambos invoquen a los constructores de la clase "Exception" correspondientes.

9. ¿Cuál es la peculiaridad de las excepciones del tipo "RuntimeException" (o de las subclases de la misma)?

10. ¿Qué información nos aporta el método "printStackTrace (): void" sobre una excepción?

11. Escribe el resultado de ejecutar el siguiente fragmento de código:

```
public static double acceso_por_indice (double [] v, int j) throws RuntimeException{
        try{
                if ((0 <= j) && (j <= v.length)){
                return v[j];
                }
                else {
                        throw new RuntimeException ("El indice " + j
                                                        + " no existe en el vector");
                }
        }
        catch (RuntimeException exc){
                throw exc;
        }
```

```
}
```

Desde el siguiente cliente "main":

```
public static void main(String [] args){

        double [] v = new double [15];
        acceso_por_indice (v, 16);
}
```

12. Escribe el resultado de ejecutar el siguiente fragmento de código:

```
public static double acceso_por_indice (double [] v, int j) throws RuntimeException{
        try{
                if ((0 <= j) && (j <= v.length)){
                return v[j];
                }
                else {
                        throw new Exception ("El indice " + j
                                                        + " no existe en el vector");
                }
        }
        catch (RuntimeException exc){
                throw exc;
        }
}
```

Desde el siguiente cliente "main":

```
public static void main(String [] args){

        double [] v = new double [15];
        acceso_por_indice (v, 16);
}
```

13. Escribe el resultado de ejecutar el siguiente fragmento de código:

```
public static double acceso_por_indice (double [] v, int j) throws Exception{
        try{
                if ((0 <= j) && (j <= v.length)){
                return v[j];
                }
                else {
                        throw new Exception ("El indice " + j
                                                        + " no existe en el vector");
                }
        }
        catch (Exception exc){
                throw exc;
        }
}
```

Desde el siguiente cliente "main":

```
public static void main(String [] args){

        double [] v = new double [15];
        acceso_por_indice (v, 16);
}
```

14. Escribe el resultado de ejecutar el siguiente fragmento de código:

```
public static double acceso_por_indice (double [] v, int j) throws Exception{
        try{
                if ((0 <= j) && (j <= v.length)){
                return v[j];
                }
                else {
                        throw new RuntimeException ("El indice " + j
                                                        + " no existe en el vector");
                }
        }
        catch (RuntimeException exc){
                throw exc;
        }
}
```

Desde el siguiente cliente "main":

```
public static void main(String [] args){

        double [] v = new double [15];
        acceso_por_indice (v, 16);
}
```

15. ¿Cuál sería la salida por pantalla del siguiente programa?

```
class Uno{
        private static int metodo ( ) {
                int valor =0;
                try {
                        valor = valor +1;
                        valor = valor + Integer.parseInt("42");
                        valor = valor + 1;
                        System.out.println ("Valor al final del try : " + valor ) ;
                }
                catch (NumberFormatException e ) {
                        valor = valor + Integer.parseInt("42");
                        System.out.println ("Valor al final del catch : " + valor) ;
                }
                finally {
                        valor = valor + 1;
                        System.out.println ("Valor al final de finally : " + valor) ;
                }
                valor = valor + 1;
                System.out.println ("Valor antes del return : " + valor);
```

```
            return valor;
        }

        public static void main (String [ ] args) {
            try {
                System.out.println (metodo ( ));
            }
            catch (Exception e) {
                System.out.println ("Excepcion en metodo ( ) " );
                e.printStackTrace ( );
            }
        }
}
```

16. ¿Cuál sería la salida por pantalla del siguiente programa?

```
class Uno{
        private static int metodo ( ) {
            int valor =0;
            try {
                valor = valor +1;
                valor = valor + Integer.parseInt("W");
                valor = valor + 1;
                System.out.println ("Valor al final del try : " + valor ) ;
            }
            catch (NumberFormatException e ) {
                valor = valor + Integer.parseInt("42");
                System.out.println ("Valor al final del catch : " + valor) ;
            }
            finally {
                valor = valor + 1;
                System.out.println ("Valor al final de finally : " + valor) ;
            }
            valor = valor + 1;
            System.out.println ("Valor antes del return : " + valor);
            return valor;
        }

        public static void main (String [ ] args) {
            try {
                System.out.println (metodo ( ));
            }
            catch (Exception e) {
                System.out.println ("Excepcion en metodo ( ) " );
                e.printStackTrace ( );
            }
        }
}
```

17. ¿Cuál sería la salida por pantalla del siguiente programa?

```
class Uno{
        private static int metodo ( ) throws NumberFormatException{
                int valor =0;
                try {
                        valor = valor +1;
                        valor = valor + Integer.parseInt("W");
                        valor = valor + 1;
                        System.out.println ("Valor al final del try : " + valor ) ;
                }
                catch (NumberFormatException e ) {
                        valor = valor + Integer.parseInt("42");
                        System.out.println ("Valor al final del catch : " + valor) ;
                        throw e;
                }
                finally {
                        valor = valor + 1;
                        System.out.println ("Valor al final de finally : " + valor) ;
                }
                valor = valor + 1;
                System.out.println ("Valor antes del return : " + valor);
                return valor;
        }

        public static void main (String [ ] args) {
                try {
                        System.out.println (metodo ( ));
                }
                catch (Exception e) {
                        System.out.println ("Excepcion en metodo ( ) " );
                        e.printStackTrace ( );
                }
        }
}
```