Java

Tema 6: Uso avanzado de clases

Ampliación de clases

Contenidos

1. Atributos y métodos estáticos
2. Envolventes de tipo (wrapper class)
3. Excepciones

Java

Atributos y métodos estáticos

El modificador static

- Los atributos y métodos de una clase precedidos de la palabra static se denominan elementos de clase.
- Los elementos static son compartidos entre todos los objetos de la clase.
- Los elementos static se pueden usar sin crear un objeto de la clase.

NombreDeClase.metodoStatic()

NombreClase.atributostatic

- Los métodos static tiene varias restricciones:
  - Sólo pueden llamar a otros métodos static
  - No pueden tener la referencia this

El modificador static: ejemplo

```
public class Coche{
   public String matricula, modelo;
   public int num_serie;
   public static int total_coches;

   public Coche(){
       total_coches++; //sirve como contador de numero de objetos
                       // creados de coche
       num_serie = total_coches; // aprovechamos para almacenar
                                 // ese valor en otra variable
   }
}

public class GestionCoches {
  public static void main (String args[ ]) {
    Coche c1 = new Coche();
    Coche c2 = new Coche();
    System.out.println("Total de coches: " + Coche.total_coches);
    System.out.println("Nº serie c1: " + c1.num_serie);
    System.out.println("Nº serie c2: " + c2.num_serie);
  }
}
```

5

El modificador static

Ejemplo:

```
public class Matematicas {
  public static double sumar(double a,double b) {
      return a+b;
  }
  public static double restar(double a,double b) {
      return a-b;
  }
}

public class UsoDeStatic {
  public static void main (String args[ ]) {
     // se pueden usar sin crear un objeto
    System.out.println("suma= " + Matematicas.sumar(3.0,2.0) );
    System.out.println("restar= "+ Matematicas.restar(6.0,3.0)
);
  }
}
```

6

El modificador static

¿Cuándo tiene sentido usar métodos `static`?

Cuando hacemos clases que no tienen estado (atributos) y tiene métodos sueltos que no comparten información (ejemplo Math)
O
Cuando solo necesitamos una instancia de la clase (ejemplo System)

Ejemplo:

java.lang.Math → Math.pow(b,e)
java.lang.System → System.getProperty(String)

7

Java

Envolventes de tipo

(wrapper class)

8

Envolventes de tipo

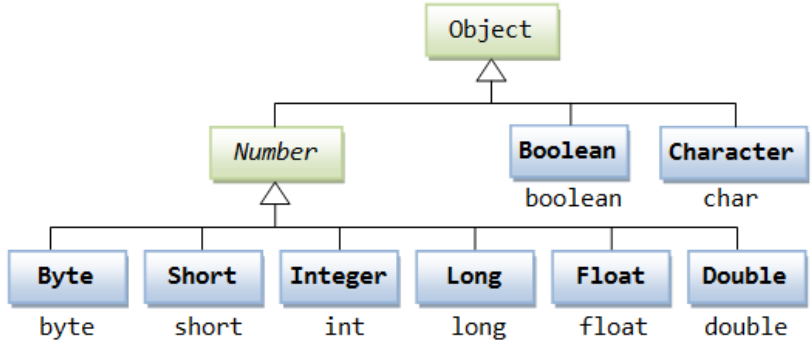
- Las envolventes de tipo son clases del api de java, que **encapsulan un tipo primitivo dentro de un objeto**.
- Se encuentran en el package java.lang, no es necesario hacer import.
- Las envolventes de tipo son:

Character, Boolean, Byte, Short, Integer, Long, Float y **Double**.

Tipo primitivo	Objetos
`int i = 6;`	`Integer i1 = 6;`
	`Integer i2 = new Integer("6");`
`long l = 34;`	`Long l1 = 34L;`
`float f = 3.7f;`	`Float f1 = 3.7f;`
`double d = 65.45;`	`Double d1 = 65.45;`
`char c = 'a';`	`Character c1 = 'a';`

9

Envolventes de tipo



10

Envolventes de tipo

- ¿Para que se usan las envolventes de tipos?

  1) Para **convertir** cadenas de texto a tipos primitivos y viceversa

  2) Para poder **pasar cualquier tipo de datos numérico** a un método

  Ejemplo: Métodos de suma con todas las combinaciones de tipos de datos

```
int sumar(int a,int b);

long sumar(long a, long b);
long sumar(long a, int b);
long sumar(int a, long b);

float sumar(float a, float b);

double sumar(double a, double b);
double sumar(double a, float b);
double sumar(float a, double b);
.....
float sumar(float a, int b);
float sumar(float a, long b);
```


Envolventes de tipo

- ¿No sería más cómodo hacer esto?

```
Number sumar(Number a,Number b);
```

  Number es una clase de la que heredan todos las clases envolventes numéricas (Integer, Long, Float, Double…)

- Es posible, pero la función anterior viene a decir que se puede pasar cualquier objeto con tal de que sea un:

```
Integer
Long
Float
Double
```

Envolventes de tipo

- El Autoboxing es una funcionalidad de Java que convierte automáticamente una variable cuyo tipo de datos es primitivo a su envolvente correspondiente, y viceversa.

```
int num1 = 4;
Integer num2 = new Integer(5);

int     resultadoA = num2;//es Integer y se convierte a int
Integer resultadoB = num1;//es int y se convierte a Integer

                  (int)   (Integer)
   resultadoA =   num1 + num2;
   resultadoB =   num1 + num2;
```


Envolventes de tipo

- Métodos mas utilizados

Método	Descripcion
String **toString**()	Convierte el valor almacenado a String
int **intValue**() double **doubleValue**() float **floatValue**() …..	Convierte el valor almacenado al tipo de datos primitivo indicado(int, double, float)
boolean **equals**(Object o)	Indica si el valor contenido es igual al parámetro o
static Integer **valueOf**(String i) statit Double **valueOf**(String i) ……	Convierte el parámetro String a objeto Integer Convierte el parámetro String a objeto Double
static int **parseInt**(String i) static double **parseDouble**(String i) …..	Convierte el parámetro String a un int Convierte el parámetro String a un double

Envolventes de tipo: ejemplos conversión

```
String cadena = "54321";
Integer num1 = Integer.valueOf(cadena);
int     num2 = Integer.parseInt(cadena);
double  num3 = Double.parseDouble(cadena);


String cadena = "123.55";
Double  num4 = Double.valueOf(cadena);
double  num5 = Double.parseDouble(cadena);

Integer n =  12
String  texto   = n.toString();
int     entero  = n.shortValue();
double  decimal = n.doubleValue();
```

15

Envolventes de tipo: clase Character

- Métodos static mas utilizados de la clase Character:

```
boolean digito = Character.isDigit('2');
boolean digito2 = Character.isDigit('a');

boolean letra = Character.isLetter('b');
boolean letra2 = Character.isLetter('2');

System.out.println(Character.toUpperCase('a'));
```

16

Java

Excepciones

17

Excepción

- Una **excepción** es un objeto que se genera automáticamente cuando se produce un error durante la ejecución de un programa, y puede ser previsto y controlado.
- Las excepciones permiten:
  - Encapsular los errores dentro de las clases
  - Separar el flujo de ejecución normal del tratamiento de errores.
- Las excepciones pueden ser:
  - tratadas-capturadas (try-catch)
  - lanzadas (throw) en los programas.

18

Tipos de excepción

- **Checked:**
  - Su tratamiento es obligatorio y el compilador así lo comprueba.
  - Todas aquellas clases descendientes de Exception, menos RuntimeException
  - *Ejemplo:* FileNotFoundException generada por la clase FileInputStream
- **Unchecked:**
  - Su tratamiento NO es obligatorio y el compilador no lo comprueba.
  - Todas aquellas clases descendientes de RuntimeException

Captura y manejo de excepciones

- Se capturan y manejan mediante los bloques try… catch:

```
try {
    // código  que puede generar una excepción
} catch (Exception ex) {
    // Tratamiento de la excepción
}
```

Captura y manejo de excepciones

- Tratamiento de una excepción.

  Ejemplo: Se puede dar una excepción si b=0 y a=0

```
public double dividir(int a,int b) {
  try {
      return a/b;
  } catch (ArithmeticException ex) {
    if (a>0) {
      return Double.POSITIVE_INFINITY;
    } else if (a<0) {
      return Double.NEGATIVE_INFINITY;
    } else {
     // lanzamos otra excepción
      throw new RuntimeException("Indeterminación 0/0",ex);
    }
   }
}
```

Ya no se puede tratar 0/0 porque es una indeterminación, así que lanzamos otra excepción

Captura y manejo de excepciones

- Si no sabemos como tratar una excepción lo mejor es relanzarla con el tipo RuntimeException

```
try  {
// código que puede generar excepción
} catch (Exception ex ) {
    //relanzamos la excepción
    throw new RuntimeException(ex);
}
```

Creacion de excepciones

- Podemos crear nuestras propias excepciones

```
public class MiExcepcion extends RuntimeException{
  public MiExcepcion(){ super("Error generado"); }
  public MiExcepcion(String mensaje){
      super("codigo " + mensaje + " no permitido");
  }
}
```

- Lanzar nuestra excepción

```
public class Validar {
  public static void esValido(String codigo) {
    if (codigo.equals("B100") || codigo.equals("C200") )
        throw new MiError(codigo);
  }
}
```


Creacion de excepciones

- Controlar y capturar la excepción que nosotros hemos lanzado

```
public class Test {
  public static void main ( String args[ ] ) {
    String codigo="B100";
    try {
        Validar.esValido(codigo);
    } catch (MiExcepcion  e) {
        System.out.println(e.getMessage());
    }
  }
}
```