

Lenguajes, Tecnologías y Paradigmas de la programación (LTP)

Práctica 2: Clases envoltorio y Genericidad



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Sergio Pérez
serperu@dsic.upv.es

Clases Envoltorio (Wrapper Classes)

Hay **2 tipos de datos** en Java

- Tipos Primitivos (**PrimitiveTypes**): `int`, `boolean`, `double`, `float`, `short` ...

Clases Envoltorio (Wrapper Classes)

Hay **2 tipos de datos** en Java

- Tipos Primitivos (**PrimitiveTypes**): `int`, `boolean`, `double`, `float`, `short` ...
- Tipos Clase (**ClassTypes**) : `Object`, `Figure`, `Triangle`, `Rectangle`, `FiguresGroup`...

Clases Envoltorio (Wrapper Classes)

Hay **2 tipos de datos** en Java

- Tipos Primitivos (**PrimitiveTypes**): `int`, `boolean`, `double`, `float`, `short` ...
- Tipos Clase (**ClassTypes**) : `Object`, `Figure`, `Triangle`, `Rectangle`, `FiguresGroup`...

A veces, nos interesaría que esos **PrimitiveTypes** fueran clases para poder utilizar algunas funcionalidades

```
Object o = 10;  
if (o instanceof double) ... // Error!! double no es una clase en Java
```

Clases Envoltorio (Wrapper Classes)

Java define una serie de “**clases envoltorio**” para sus tipos Primitivos

Primitive Data Type	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Clases Envoltorio (Wrapper Classes)

¿Cómo convierto de `int` a `Integer`?

Clases Envoltorio (Wrapper Classes)

¿Cómo convierto de `int` a `Integer`?

```
int i = 10;
```

```
Integer iE = i;
```

```
Integer iE = 10;
```

Clases Envoltorio (Wrapper Classes)

¿Cómo convierto de `int` a `Integer`?

```
int i = 10;
```

```
Integer iE = i;
```

```
Integer iE = 10;
```

¿Cómo convierto de `Integer` a `int`?

```
Integer iE = new Integer(10);
```

```
int i = iE;
```

```
int i = new Integer(10);
```

En ambos casos la conversión sucede
de manera implícita

Ejercicio 2

Objetivo:

Saber usar las clases *wrapper*, comprender la sintaxis usada

¿Como?:

Usando los constructores de las clases envoltorio

NOTA: Haced solo 3 tipos: **Integer**, **Double** y **Character**

Genericidad

```
public class AlmacenString{  
    String element;  
  
    public AlmacenString(String value){  
        element = value;  
    }  
    public String get(){  
        return element;  
    }  
    public void set(String value){  
        element = value;  
    }  
}
```

Genericidad

```
public class AlmacenString{
    String element;

    public AlmacenString(String value){
        element = value;
    }
    public String get(){
        return element;
    }
    public void set(String value){
        element = value;
    }
}
```

```
public class AlmacenInteger{
    Integer element;

    public AlmacenInteger(Integer value){
        element = value;
    }
    public Integer get(){
        return element;
    }
    public void set(Integer value){
        element = value;
    }
}
```

Genericidad

```
public class AlmacenString{  
    String element;  
  
    public AlmacenString(String value){  
        element = value;  
    }  
    public String get(){  
        return element;  
    }  
    public void set(String value){  
        element = value;  
    }  
}
```

```
public class AlmacenInteger{  
    Integer element;  
  
    public AlmacenInteger(Integer value){  
        element = value;  
    }  
    public Integer get(){  
        return element;  
    }  
    public void set(Integer value){  
        element = value;  
    }  
}
```

Genericidad

```
public class AlmacenString{  
    String element;  
  
    public AlmacenString(String value){  
        element = value;  
    }  
    public String get(){  
        return element;  
    }  
    public void set(String value){  
        element = value;  
    }  
}
```

```
public class AlmacenInteger{  
    Integer element;  
  
    public AlmacenInteger(Integer value){  
        element = value;  
    }  
    public Integer get(){  
        return element;  
    }  
    public void set(Integer value){  
        element = value;  
    }  
}
```

Genericidad

```
public class Almacen{  
    T element;  
  
    public Almacen(T value){  
        element = value;  
    }  
    public T get(){  
        return element;  
    }  
    public void set(T value){  
        element = value;  
    }  
}
```

Genericidad

```
public class Almacen{  
    T element;  
  
    public Almacen(T value){  
        element = value;  
    }  
    public T get(){  
        return element;  
    }  
    public void set(T value){  
        element = value;  
    }  
}
```

```
public class Almacen<T>{  
    T element;  
  
    public Almacen(T value){  
        element = value;  
    }  
    public T get(){  
        return element;  
    }  
    public void set(T value){  
        element = value;  
    }  
}
```

Genericidad

```
public static void main(String[] args)
{
    Almacen<String> almacenString = new Almacen<String>("Value");
}
```

```
public class Almacen<T>{
    T element;

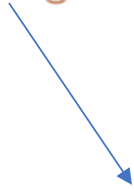
    public Almacen(T value){
        element = value;
    }
    public T get(){
        return element;
    }
    public void set(T value){
        element = value;
    }
}
```


Genericidad

```
public static void main(String[] args)
{
    Almacen<String> almacenString = new Almacen<String>("Value");
}

public class Almacen<T>{
    T element;

    public Almacen(T value){
        element = value;
    }
    public T get(){
        return element;
    }
    public void set(T value){
        element = value;
    }
}
```



Genericidad

```
public static void main(String[] args)
{
    Almacen<String> almacenString = new Almacen<String>("Value");
}
```

```
public class Almacen<T>{
    T element;

    public Almacen(T value){
        element = value;
    }
    public T get(){
        return element;
    }
    public void set(T value){
        element = value;
    }
}
```



```
public class Almacen{
    String element;

    public AlmacenString(String value){
        element = value;
    }
    public String get(){
        return element;
    }
    public void set(String value){
        element = value;
    }
}
```

Genericidad

¿Cómo instancio una estructura de datos genérica `ArrayList<E>`?

Genericidad

¿Cómo instancio una estructura de datos genérica `ArrayList<E>`?

- 1) ¿De qué tipo quiero hacer mi `ArrayList`?
`String`

Genericidad

¿Cómo instancio una estructura de datos genérica `ArrayList<E>`?

1) ¿De qué tipo quiero hacer mi `ArrayList`?

`String`

2) Declaro un `ArrayList` de tipo `String`

```
ArrayList<String> myArray
```

Genericidad

¿Cómo instancio una estructura de datos genérica `ArrayList<E>`?

1) ¿De qué tipo quiero hacer mi `ArrayList`?

`String`

2) Declaro un `ArrayList` de tipo `String`

```
ArrayList<String> myArray
```

3) Llamo al constructor de `ArrayList` de `String`

```
ArrayList<String> myArray = new ArrayList<String>();
```

Ejercicio 3

Objetivo:

Saber usar las clases predefinidas genéricas (Clase `ArrayList<E>`)

¿Como?:

Completando la clase `ArrayListUse` para imprima en pantalla las líneas de un fichero ordenadas alfabéticamente

A tener en cuenta:

- 1) Dentro del try debes incluir (solo) la instanciación del `Scanner`
- 2) Recuerda cerrar el objeto `Scanner` tras el recorrido (`file.close()`)
- 3) `Collections.sort(list)` ordena una lista de `String` automáticamente

NOTA: Para probar tu implementación descarga el fichero `textoEjercicio3.txt` y muévelo a la raíz del proyecto BlueJ (la carpeta donde están los paquetes `practica1` y `practica2`)

Ejercicio 5

Objetivo:

Definir una clase genérica **QueueAC<T>** que implemente el comportamiento de una cola mediante un **Array Circular**

¿Como?:

- Completa la clase **QueueAC** proporcionada con los atributos necesarios e implementa los métodos declarados
- Prueba tu implementación ejecutando la clase **QueueApp**

A tener en cuenta:

Debes invocar el método **increase(int i)** para aumentar los índices de manera circular.

Ejercicio 6

Objetivo:

Definir una clase genérica **QueueAL<T>** que implemente el comportamiento de una cola mediante un **ArrayList**

¿Como?:

- Completa la clase **QueueAL** proporcionada con los atributos necesarios e implementa los métodos declarados
- Prueba tu implementación ejecutando la clase **QueueApp** con las modificaciones necesarias

A tener en cuenta:

Debes usar los métodos disponibles en la clase **ArrayList**

Ejercicio 7

Objetivo:

Introducir la idea de que podemos restringir la genericidad mediante el uso de la herencia

¿Qué tipo de objetos representa esta nueva clase?

```
public class FiguresQueue<T extends Figure> extends QueueAL<T> {}
```

Ejercicio 8

Objetivo:

Restringir la genericidad mediante herencia y aprovechar lo que ésta nos proporciona.

```
public class FiguresQueue<T extends Figure> extends QueueAL<T> {}
```

¿Como?:

- Utilizando el método **area** definido en *Figure*

A tener en cuenta:

Debes visitar todos los elementos del *ArrayList* que hay en *QueueAL* sin eliminarlos.