Clase 15: Genéricos

Programación & Laboratorio III

Agenda

- **Ejemplo**
- Genericidad
- Declaración de tipos genéricos
- Declaración de una clase genérica
- Tipos genéricos - Convención
- Ejemplos

Ejemplo

```
public class Caja {

     private List<Object> elementos = new ArrayList<>();
     private int tope;

     private Caja(int tope) {
        this.tope = tope;
     }

     public boolean agregarElemento(Object o) {
        if (tope < elementos.size()) {
            elementos.add(o);
            return true;
        }
        return false;
     }
}
```

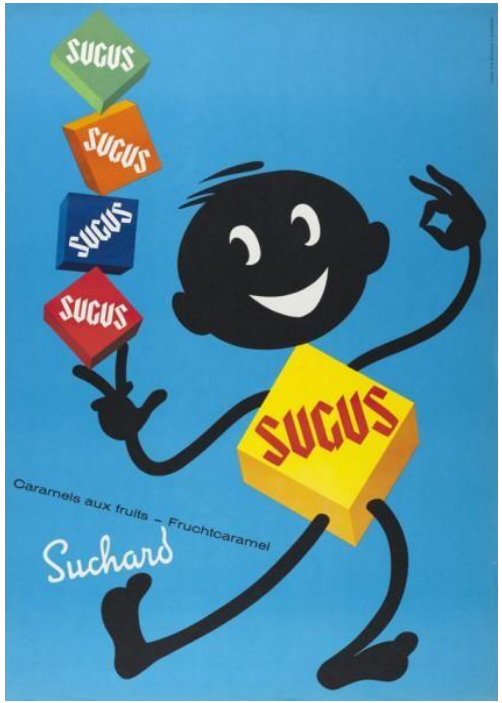
Ejemplo (2)

```
public class Caramelo {

     private String marca;
     private String sabor;

     private Caramelo(String marca, String sabor) {
        this.marca = marca;
        This.sabor = sabor;
     }

    @Override
    public String toString() {
       return “Caramelo= [marca= ” + marca +
            “, sabor= “ + sabor + “]”);
    }
}
```



Ejemplo (3)

```
public class Main {

     public static void main(String[] args) {
        Caja miCaja = new Caja(10);

        Caramelo c1 = new Caramelo(“Sugus”, “menta”);

        miCaja.agregarElemento(c1);

        Perro p1 = new Perro();
        Gato g1 = new Gato();

        miCaja.agregarElemento(p1);
        miCaja.agregarElemento(g1);

        ...
```

Ejemplo (4)

```
        ...

    //Imprimir los caramelos de la caja
    for (Object o : miCaja.getElementos()) {
        if (o instanceOf Caramelo) {
            System.out.println(((Caramelo) o).toString());
        }
    }
}
```

Conclusión

- El uso de Object como una referencia genérica es potencialmente inseguro y no se puede hacer nada para que el programador no cometa un error equivocado.
- El error es descubierto en tiempo de ejecución, al momento de realizar el casteo cuando se lanza una excepción del tipo `ClassCastException`.

Solución

```
public class Caja<T> {

     private List<T> elementos = new ArrayList<>();
     private int tope;

     private Caja(int tope) {
        this.tope = tope;
     }

     public boolean agregarElemento(T t) {
        if (tope < elementos.size()) {
            elementos.add(t);
            return true;
        }
        return false;
     }
}
```

Solución (2)

```
public class Main {

     public static void main(String[] args) {
        Caja<Caramelo> miCaja = new Caja(10);

        Caramelo c1 = new Caramelo(“Sugus”, “menta”);
        miCaja.agregarElemento(c1);

        Perro p1 = new Perro();
        miCaja.agregarElemento(p1);

        //Imprimir los caramelos de la caja
        for (Caramelo c : miCaja.getElementos()) {
           System.out.println(c.toString());
        }
     }
}
```

Error en tiempo de compilación

Agenda

- ~~Ejemplo~~
- **Genericidad**
- Declaración de tipos genéricos
- Declaración de una clase genérica
- Tipos genéricos - Convención
- Ejemplos

Genericidad

- El término “genericidad” se refiere a una serie de técnicas que permiten escribir algoritmos o definir contenedores de forma que puedan aplicarse un amplio rango de tipos de datos.
- Es una construcción importante en los lenguajes de programación orientada a objetos, que si bien no es exclusiva de este tipo de lenguajes, ha adquirido verdadera importancia y uso.
- Permite definir una clase o un método sin especificar el tipo de dato o parámetros, de esta forma se puede cambiar la clase para adaptarla a diferentes usos sin tener que reescribirla.

Genericidad (2)

- La razón de la genericidad se base principalmente en el hecho de que los algoritmos de resolución de numerosos problemas no dependen del tipo de dato que procesan. Por ejemplo: un algoritmo que implementa una pila de caracteres es igual al que se usa para implementar una pila de números enteros.
- La programación genérica significa escribir un código que puedan reutilizar muchos tipos diferentes de objetos

Agenda

- ~~Ejemplo~~
- ~~Genericidad~~
- **Declaración de tipos genéricos**
- Declaración de una clase genérica
- Tipos genéricos - Convención
- Ejemplos

Declaración de tipos genéricos

- Una declaración de tipos genéricos puede tener múltiples tipos parametrizados, separados por comas.
- Todas las invocaciones de clases genéricas son expresiones de una clase. Al instanciar una clase genérica no se crea una nueva clase.
- No se puede usar el tipo genérico <T> como tipo de un campo estático o en cualquier lugar dentro de un método estático o inicializador estático. Por ejemplo:

```
private static final T MI_CONSTANTE = new T();
```

Declaración de tipos genéricos (2)

- No se puede usar un tipo de datos genérico en la creación de objetos y arreglos. Por ejemplo:

```
public class Main {

     public static void main(String[] args) {
        T[] miArreglo = new T[10];

        T t = new T();
        miArreglo[0] = t;

        //Imprimir los elementos del arreglo
        for (int i=0; i<10; i++) {
           System.out.println(t.toString());
        }
     }
}
```

Declaración de tipos genéricos (3)

- Dentro de una definición de clases, el tipo genérico puede aparecer en cualquier declaración no estática donde se podría utilizar cualquier tipo de datos concreto.

```
public class Caja<T> {

     private List<T> elementos = new ArrayList<>();
      …

     public boolean agregarElemento(T t) {
      ...

     }
}
```

Agenda

- ~~Ejemplo~~
- ~~Genericidad~~
- ~~Declaración de tipos genéricos~~
- **Declaración de una clase genérica**
- Tipos genéricos - Convención
- Ejemplos

Declaración de una clase genérica

- Una clase genérica o parametrizable es una clase con una o más variables de tipo genérico.

```
public class MiClase<tipo_genérico> {

            ...
}
```

Donde “MiClase” es el nombre de la clase genérica y “tipo_genérico” es el tipo parametrizado genérico.

Agenda

- ~~Ejemplo~~
- ~~Genericidad~~
- ~~Declaración de tipos genéricos~~
- ~~Declaración de una clase genérica~~
- **Tipos genéricos - Convención**
- Ejemplos

Tipos genéricos - Convención

- E → elemento de una colección
- K → clave
- N → número
- T → tipo
- V → valor
- S, U, V, etc. → para segundos, terceros y cuartos tipos.

Agenda

- ~~Ejemplo~~
- ~~Genericidad~~
- ~~Declaración de tipos genéricos~~
- ~~Declaración de una clase genérica~~
- ~~Tipos genéricos - Convención~~
- **Ejemplos**

Ejemplos

- Ejemplo 1:

```
private Box<Integer> numeros = new Box<>();
```

- Ejemplo 2:

```
public interface Par<K, V> {

  ...

}

public class ParOrdenado<K, V> implements Pair<K,V> {

}
```

Ejemplos (2)

- Ejemplo 3:

```
public class NumeroNatural<T extends Integer> {

 ...

}
```

- Ejemplo 4:

```
public class A {...}

public interface B {...}

public class C <T extends A & B> {...}
```

Bibliografía oficial

- https://docs.oracle.com/javase/tutorial/java/generics/types.html