Clases y métodos genéricos

Programación I

Grado en Ingeniería Informática

MDR, JCRdP y JDGD

Genericidad

- Permite que una clase, interfaz o método se defina para usar referencias a una clase no especificada
- Se introdujo en la versión 5 de Java
- Al usar un elemento genérico se establece qué clase concreta se está usando
- La comprobación del uso de la clase correcta se hace durante la compilación
- Estos chequeos evitan la necesidad de usar operaciones de cast
- El principal uso de la genericidad son los contenedores

Ejemplo Caja (1 /4)

```
public class Caja{ //Caja no genérica
    private Object objeto;
    //Atributo Object =>cualquier objeto
    public void estableceObjeto(Object o){
        object=o;
    }
    public Object dameObjeto(){
        return objeto;
    }
}
```

Ejemplo Caja (2/4)

- La clase Caja almacena una referencia a **Object**
- Por el polimorfismo puede almacenar objetos de cualquier clase
- Recuperar la referencia original requiere un cast

```
Caja c = new Caja();
c.estableceObjeto("Ristra");
String s = (String) c.dameObjeto();
```

Ejemplo Caja (3/4)

```
public class Caja<T>{ //Caja genérica
    private T objeto;
    public void estableceObjeto(T o){
        object=o;
    }
    public T dameObjeto(){
        return objeto;
    }
}
```

Ejemplo Caja (4/4)

- Con la genericidad podemos establecer de qué tipo queremos la caja
- Se define con Caja<Tipo>
- Recuperar la referencia no requiere cast
- No podemos almacenar otro tipo del establecido (error de compilación)

```
Caja<String> c = new Caja<String>();
c.estableceObjeto("Ristra");
String s = c.dameObjeto();
```

Convención de nombres

- Por convención los nombres de parámetros genéricos son letras únicas en mayúsculas
- Los nombres más comunes son:
  - E – Elemento de un contenedor
  - K – Clave de un contenedor
  - N – Número
  - T – Tipo
  - V – Valor

Más sobre genericidad

- Es posible usar varios parámetros, en cuyo caso, se separan por "," tanto, en la definición, como en el uso con los tipos reales
- Desde la versión 7 de Java, cuando se llama al constructor de una clase genérica es posible usar solo el diamante "<>" ya que el compilador es capaz de inferir el tipo del contexto en que se usa

```
public class Pareja<A,B>{
  public Pareja(A a, B b){ ... }
  ...
}
...
Pareja<String, Integer> par= new Pareja<>("",0);
```

Genericidad y subclases

- Que una clase A sea subclase de otra B no quiere decir que una clase genérica G<A> sea subclase de G<B>
- Para aceptar genericidad de cualquier tipo se usa el comodín "?".
- Se pueden establecer límites con "? extends A" indicando que sólo se admite A y sus clases derivadas

```
static int cuenta(List<? extends Number> l){
}
```

Limitaciones

Para `class clase<T>{...}`

- T no puede ser un tipo primitivo
- No es posible crear objetos paramétricos usando `new T()`
- No se pueden declarar atributos estáticos del tipo T
- No se puede usar cast o instanceof de tipos paramétricos
- No se pueden crear arrays de tipos paramétricos

Bibliografía

- **Lesson: Generics** http://docs.oracle.com/javase/tutorial/java/generics/
- **Lesson: Generics** ***by Gilad Bracha*** http://docs.oracle.com/javase/tutorial/extra/generics/index.html