Computación y Estructuras Discretas I

Andrés A. Aristizábal P.
aaaristizabal@icesi.edu.co

Departamento de Computación y Sistemas Inteligentes

2024-2

Agenda del día

1. **Generics**
   - Control de lectura
   - Introducción
   - Tipos genéricos
   - Tipos raw
   - Métodos genéricos
   - Wildcards
   - Lo no permitido por Generics
   - Ejercicio

Control de lectura

Instrucciones

1. Ingresar a https://www.socrative.com
2. Log in
3. Student Login
4. Room name: FTJF5SP
5. Nombre
6. Comenzar

Agenda del día

1. **Generics**
   - Control de lectura
   - Introducción
   - Tipos genéricos
   - Tipos raw
   - Métodos genéricos
   - Wildcards
   - Lo no permitido por Generics
   - Ejercicio

Introducción

¿Por qué Generics?

Introducción

¿Por qué Generics?

- Generics permite que, al definir una clase, interfaz o método, se puedan utilizar tipos (clases o interfaces) como parámetros.
- Busca independizar el proceso del tipo de datos sobre los que se aplica.
- Definición de modelos generales.

Introducción

¿Por qué Generics?

- Generics permite que, al definir una clase, interfaz o método, se puedan utilizar tipos (clases o interfaces) como parámetros.
- Busca independizar el proceso del tipo de datos sobre los que se aplica.
- Definición de modelos generales.

¿Y esto que acarrea?

Introducción

¿Por qué Generics?

- Generics permite que, al definir una clase, interfaz o método, se puedan utilizar tipos (clases o interfaces) como parámetros.
- Busca independizar el proceso del tipo de datos sobre los que se aplica.
- Definición de modelos generales.

¿Y esto que acarrea?

Al utilizar parámetros en la definición de clases, interfaces y métodos, se provee una nueva manera de reutilizar código.

Introducción

Entonces, ¿cuáles son los beneficios de un código que usa Generics y otro que no?

Introducción

Entonces, ¿cuáles son los beneficios de un código que usa Generics y otro que no?

- Chequeos de tipos más rigurosos en tiempo de compilación.
- Eliminación de casts.
- Implementación de algoritmos genéricos.

Ejemplo

```
ArrayList lista = new ArrayList();
lista.add("Hola");
String cadena = (String) lista.get(0);

ArrayList lista <String> = new ArrayList<String>();
lista.add("Hola");
String cadena = lista.get(0);
```

Introducción

¿Se pueden utilizar tipos primitivos al utilizar Generics?

Introducción

¿Se pueden utilizar tipos primitivos al utilizar Generics?

- No se pueden utilizar con primitivos pero sí con las clases que corresponden con ellos.
- No se puede hacer un genérico tipo int pero si un Integer.
- Autoboxing a las clases wrappers.

Introducción

¿Qué relación puede existir entre sobrecarga de métodos y generics?

Introducción

¿Qué relación puede existir entre sobrecarga de métodos y generics?

- Si las operaciones realizadas por varios métodos sobrecargados son idénticas, pueden codificarse de manera compacta utilizando un método genérico.
- Con base en los tipos de los argumentos que se pasan al método genérico, el compilador maneja cada llamada al método de manera apropiada.
- Además de establecer las llamadas a los métodos, el compilador determina si las operaciones en el cuerpo del método se pueden aplicar a los elementos del tipo almacenado en el argumento de la clase genérica (arreglo, ArrayList).

Agenda del día

1. **Generics**
   - Control de lectura
   - Introducción
   - Tipos genéricos
   - Tipos raw
   - Métodos genéricos
   - Wildcards
   - Lo no permitido por Generics
   - Ejercicio

Ejemplo

```
public class Caja {
  private Object objeto;

  public void modificar(Object objeto) {
    this.objeto = objeto;
  }

  public Object dar() {
    return objeto;
  }
}
```

Tipos genéricos

¿Qué problemas se pueden presentar al utilizar la clase no genérica anterior?

Tipos genéricos

¿Qué problemas se pueden presentar al utilizar la clase no genérica anterior?

- No se puede verificar en tiempo de compilación como se utiliza la clase.
- Una parte del código puede ubicar un tipo de objeto en la Caja y esperar que se retorne un objeto de ese mismo tipo, mientras que, por error, en otra parte del código se le puede pasar otro tipo de objeto, generando un error en tiempo de ejecución.

Tipos genéricos

¿Qué es mejor, un error en tiempo de compilación o en tiempo de ejecución?

Tipos genéricos

¿Qué es mejor, un error en tiempo de compilación o en tiempo de ejecución?

- Un error en tiempo de compilación.
- Errores en tiempo de compilación son más fáciles de encontrar.

Tipos genéricos

¿Cómo se declara una clase genérica?

Tipos genéricos

¿Cómo se declara una clase genérica?

- Al momento de declarar la clase, luego del nombre de ésta, se especifican los parámetros de tipos delimitados por $< >$.
- Los objetos a ser utilizados dentro de la clase se reemplazan por el tipo de parámetro de entrada.
- Estas variables de tipo pueden ser de cualquier tipo no primitivo.
- El mismo procedimiento puede utilizarse para declarar interfaces genéricas.

Tipos genéricos

¿Cuál sería la versión genérica del ejemplo previo?

¿Cuál sería la versión genérica del ejemplo previo?

Ejemplo

```
public class Caja<T> {
  private T tipo;

  public void modificar(T tipo) {
    this.tipo = tipo;
  }

  public T dar() {
    return tipo;
  }
}
```

Tipos genéricos

¿Cómo se declara una variable de tipo genérica?

Tipos genéricos

¿Cómo se declara una variable de tipo genérica?

- `Caja<Integer> cajaEnteros;`
- `Caja<Double> cajaReales;`
- `Caja<String> cajaString;`
- `Caja< Caja<String> > cajaCajaString;`

Tipos genéricos

¿Cómo se declara una variable de tipo genérica?

- `Caja<Integer> cajaEnteros;`
- `Caja<Double> cajaReales;`
- `Caja<String> cajaString;`
- `Caja< Caja<String> > cajaCajaString;`

¿Cómo se crea un objeto de tipo genérico?

Tipos genéricos

¿Cómo se declara una variable de tipo genérica?

- `Caja<Integer> cajaEnteros;`
- `Caja<Double> cajaReales;`
- `Caja<String> cajaString;`
- `Caja< Caja<String> > cajaCajaString;`

¿Cómo se crea un objeto de tipo genérico?

- `Caja<Integer> cajaEnteros = new Caja<Integer>();`
- `Caja<Integer> cajaEnteros = new Caja<>();`

Tipos genéricos

¿Pero por qué es válido `Caja<Integer> cajaEnteros = new Caja<>();`?

Tipos genéricos

¿Pero por qué es válido `Caja<Integer> cajaEnteros = new Caja<>();`?

- Al invocar el constructor de una clase genérica se pueden reemplazar los argumentos de tipo con un conjunto vacío siempre y cuando el compilador pueda determinarlos o inferirlos a partir del contexto.
- Como este conjunto vacío se denota como $<>$ se denomina notación diamante.

Tipos genéricos

¿Puede una clase tener múltiples parámetros de tipos?

Tipos genéricos

¿Puede una clase tener múltiples parámetros de tipos?

- Sí.
- `Terna<`T_1, T_2, T_3`> terna;`

Tipos genéricos

Ejercicio

Declare la clase genérica Terna (con 7 métodos, constructor, 3 getters y 3 setters) que implemente la interface ITerna. Posteriormente declare variables de tipo Terna e instáncielas utilizando distintos argumentos de tipo.

Tipos genéricos

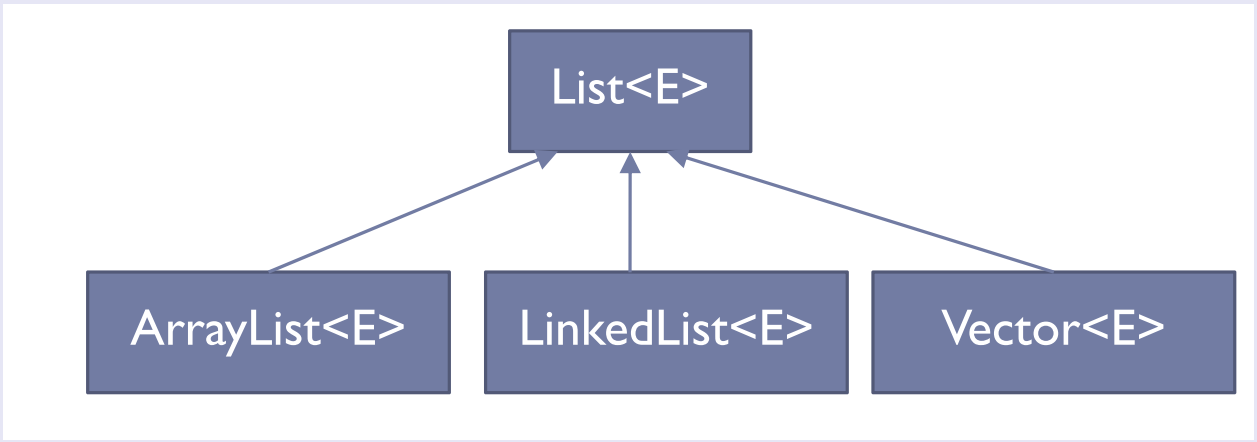
```
public interface ITerna<T1,T2,T3> {
  public T1 obtenerPrimero();
  public T2 obtenerSegundo();
  public T3 obtenerTercero();
}
public class Terna<T1,T2,T3> implements ITerna<T1,T2,T3> {
  private T1 primero;
  private T2 segundo;
  private T3 tercero;

  public Terna(T1 primero, T2 segundo, T3 tercero) {
    this.primero = primero;
    this.segundo = segundo;
    this.tercero = tercero;
  }
  public T1 obtenerPrimero() {
    return primero;
  }
  public T2 obtenerSegundo() {
    return segundo;
  }
  public T3 obtenerTercero() {
    return tercero;
  }
}
```

Tipos genéricos

¿Cuál sería una interfaz genérica estándar de Java?

Tipos genéricos

¿Cuál sería una interfaz genérica estándar de Java?



Ejemplo

```
public class TestArrayList {
  public static void main(String[] args) {
    List<Figura> lista = new ArrayList<Figura>();
    lista.add(new Circulo());
    Figura f = lista.get(0);
    Circulo c = (Circulo) lista.get(0);
  }
}
```

Tipos genéricos

¿Se pueden restringir los tipos genéricos?

Tipos genéricos

¿Se pueden restringir los tipos genéricos?

- Se pueden restringir los tipos genéricos de tal manera que se pueda trabajar con un tipo específico y sus subtipos.
- Para establecer el límite superior, se coloca después del nombre del parámetro de tipo la palabra clave extends y el nombre de la clase o interfaz que representa dicha restricción.
- El extends se utiliza para clases e interfaces.

Ejemplo

```
public class CajaNumeros<T extends Number> {
   private T dato;
   public T darElemento() {
      return dato;
   }
   public void modificarDato(T dato) {
      this.dato = dato;
   }
}
```

Agenda del día

1. **Generics**
   - Control de lectura
   - Introducción
   - Tipos genéricos
   - Tipos raw
   - Métodos genéricos
   - Wildcards
   - Lo no permitido por Generics
   - Ejercicio

Tipos raw

¿Qué es el tipo raw o crudo?

Tipos raw

¿Qué es el tipo raw o crudo?

- Es el nombre de una clase o interfaz genérica cuando no se le pasa ningún argumento de tipo.
- `Caja cajaCruda = new Caja();`
- En este caso `Caja` es un tipo raw de `Caja<T>`.
- Un tipo de clase o interfaz no genérica no es un tipo raw.

Tipos raw

¿Qué advertencias se pueden presentar al utilizar tipos raw?

Tipos raw

¿Qué advertencias se pueden presentar al utilizar tipos raw?

- Se presenta una advertencia ya que a un tipo parametrizado se le asigna un tipo raw.

```
Caja cajaCruda = new Caja();
Caja<Integer> intCaja = cajaCruda;
```

- Se presenta una advertencia pues el tipo raw se salta los chequeos de tipo genérico y así pasa su manejo de código inseguro a tiempo de ejecución.

```
Caja<String> stringCaja = new Caja<>();
Caja cajaCruda = stringCaja;
cajaCruda.modificar(8);
```

Agenda del día

1. **Generics**
   - Control de lectura
   - Introducción
   - Tipos genéricos
   - Tipos raw
   - Métodos genéricos
   - Wildcards
   - Lo no permitido por Generics
   - Ejercicio

Métodos genéricos

¿Qué son los métodos genéricos?

Métodos genéricos

¿Qué son los métodos genéricos?

- Son aquellos métodos que introducen su propio tipo de parámetros.
- Es similar a la declaración de tipo genérica, pero el alcance del parámetro está limitado al método donde se declara.
- Se admiten métodos genéricos estáticos y no estáticos, al igual que constructores genéricos.
- La sintaxis de un método genérico incluye un parámetro de tipo entre $<>$ y aparece antes del tipo de retorno del método.

Ejemplo

```java
public class GenericMethodTest {
   // generic method printArray
   public static < E > void printArray( E[] inputArray ) {
      // Display array elements
      for(E element : inputArray) {
         System.out.print(element+" ");
      }
      System.out.println();
   }

   public static void main(String args[]) {
      // Create arrays of Integer, Double and Character
      Integer[] intArray = { 1, 2, 3, 4, 5 };
      Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };
      Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };

      System.out.println("Array integerArray contains:");
      printArray(intArray);   // pass an Integer array

      System.out.println("\nArray doubleArray contains:");
      printArray(doubleArray);   // pass a Double array

      System.out.println("\nArray characterArray contains:");
      printArray(charArray);   // pass a Character array
   }
}
```

Métodos genéricos

¿Se puede utilizar el operador relacional con tipos referenciados?

Métodos genéricos

¿Se puede utilizar el operador relacional con tipos referenciados?

- No, sin embargo, es posible comparar dos objetos de la misma clase, si esa clase implementa a la interfaz genérica Comparable<T>.
- Los objetos de clases que implementan Comparable<T> tienen el método compareTo(T t).
- Todas las clases de envoltura de tipos para tipos primitivos implementan a Comparable<T>.
- Un beneficio de implementar la interfaz genérica Comparable<T> es que los objetos de clases que implementan esta interfaz pueden utilizarse con métodos de ordenamiento y búsqueda sobre las clases Collections.

Métodos genéricos

```
class GenericInsertionSorter {
  public <T extends Comparable<T>> void sort(T[] elems) {
    int size = elems.length;
    for (int outerLoopIdx = 1; outerLoopIdx < size; outerLoopIdx++) {
      for (int innerLoopIdx = outerLoopIdx; innerLoopIdx > 0; innerLoopIdx--) {
        if (elems[innerLoopIdx – 1].compareTo(elems[innerLoopIdx]) > 0) {
          T temp = elems[innerLoopIdx – 1];
          elems[innerLoopIdx – 1] = elems[innerLoopIdx];
          elems[innerLoopIdx] = temp;
        }
      }
    }
  }
}
```

Agenda del día

1. **Generics**
   - Control de lectura
   - Introducción
   - Tipos genéricos
   - Tipos raw
   - Métodos genéricos
   - Wildcards
   - Lo no permitido por Generics
   - Ejercicio

Wildcards

¿Qué son los comodines o wildcards en Generics?

Wildcards

¿Qué son los comodines o wildcards en Generics?

- El signo de interrogación ?, también denominado comodín o wildcard representa un tipo no conocido.
- Un comodín de tipo parametrizado es una instancia de un tipo genérico donde al menos un argumento de tipo es un wildcard.
- Ejemplos: `Collection<?>`, `List<? extends Number>`, `Pair<String,?>`.
- Los comodines pueden utilizarse en gran variedad de situaciones, como tipo del parámetro, campo o variable local.
- Los wildcard nunca se utilizan como un argumento de tipo para una invocación de un método genérico o una creación de una instancia de una clase genérica.

Agenda del día

1. **Generics**
   - Control de lectura
   - Introducción
   - Tipos genéricos
   - Tipos raw
   - Métodos genéricos
   - Wildcards
   - Lo no permitido por Generics
   - Ejercicio

Lo no permitido por Generics

¿Qué no permite Generics?

Lo no permitido por Generics

¿Qué no permite Generics?

- No se puede definir un miembro de una clase como estático genérico parametrizado. Cualquier intento de este tipo produce un error de compilación, pues hace una referencia estática a un tipo T no estático.
- No se pueden crear instancias de tipo T (tipo genérico).
- Generics no es compatible con tipos de datos primitivos. Aunque si se pueden utilizar las clases wrappers en lugar de los tipos de datos primitivos y luego usar los primitivos cuando se pasan los valores.
- No se puede crear una clase Excepción genérica.

Agenda del día

1. **Generics**
   - Control de lectura
   - Introducción
   - Tipos genéricos
   - Tipos raw
   - Métodos genéricos
   - Wildcards
   - Lo no permitido por Generics
   - Ejercicio

Ejercicio

Ejercicio

- Implemente una lista enlazada simple genérica.