Genericos

Java 21

JoeDayz | 2024

Generics

Generics is a feature of Java language available since Java SE 5.

- It allows operations on objects of various types while providing compile-time type safety.
- Prior to Java SE 5 (no generics style), values were wrapped within the class using the type Object.
- Post Java SE 5 (generics style), values are wrapped within the class with deferred exact type identification.
- Generic type avoids hard-coding the exact type as part of the class design.

Without Generics

```
public class Some {
  private Object value;
  public Object getValue() {
    return value;
  }
  public void setValue(Object value) {
    this.value = value;
  }
}
```

With Generics

```
public class Some<T> {
  private T value;
  public T getValue() {
    return value;
  }
  public void setValue(T value) {
    this.value = value;
  }
}
```

✤ **Note:** In the example, T is not a keyword, or class or interface name, but a generic type marker. Other markers can be used: T (type), V (value), K (key), and any other marker you like, which could be a word or even a single letter.

Use Generics

The use of Generics helps to produce compact, type-safe code.

- Without generics:
  - Any type can be assigned to a variable or parameter whose type is Object
  - Programmatic type-check using the `instanceof` operator is required to ensure that you don't accidentally cast variable to the wrong type
- With generics:
  - Compiler checks that the type that is assigned, or passed as parameter, corresponds to the generic type declaration, rejecting code that attempts to use types that don't match
  - No programmatic type-check or type-casting is required

Without Generics

```
Some some = new Some();
some.setValue(new Product("Tea",1.99));
some.setValue("something");
Object value = some.getValue();
if (value instanceof Product) {
  Product product = (Product)value;
}
if (value instanceof String) {
  String text = (String)value;
}
```

With Generics

```
Some<Product> some = new Some<>();
some.setValue(new Product("Tea",1.99));
some.setValue("something");
Product product = some.getValue();
```

Note: Generics can be used with both classes and interfaces. Many existing Java interfaces utilize generics.