Programming Software Systems

Lab 10

Agenda

- Revision of generics in Java: from different angle
- Warm-up exercises
- Solving Problems
- Q&A

Learning outcome:

- Strengen the knowledge generic in Java
- Application of generics in real-world problems
- Code-review & improving code quality

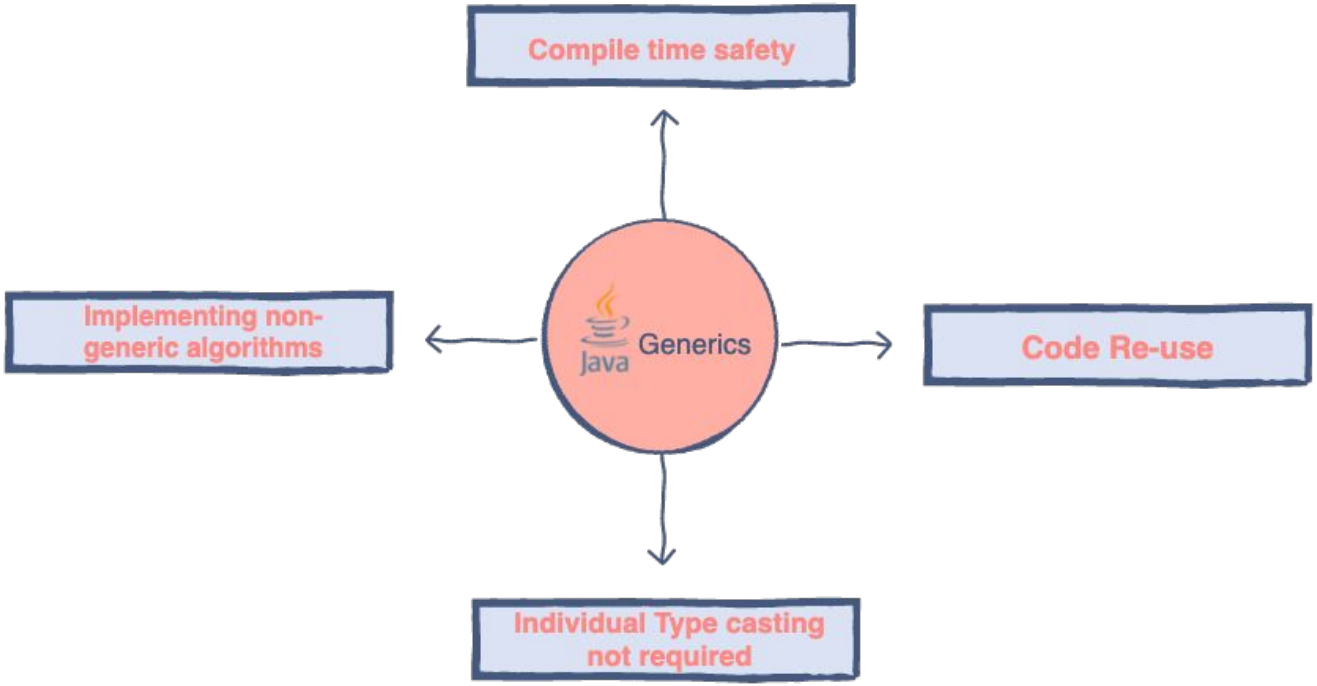
Java generics

https://data-flair.training/blogs/java-generics/

What is Java Generics?

Generics in Java is like templates in C++. The thought is to permit write (Integer, String and so on and user-defined types) to be a parameter to techniques, classes, and interfaces. For instance, classes like HashSet, ArrayList, HashMap, and so forth utilize generics exceptionally well. We can utilize them for any kind.

What is Java Generics?



Java Generics in class

<> is used to specify parameter types in Java generic class creation.

The syntax for creating a Java generic class

```
BaseType <Type> obj = new BaseType <Type>()
```

Java Generics in class

```
class Test<T>
   {
// An object of type T is declared
       T obj;
       Test(T obj)
       {  this.obj = obj;  } // constructor
       public T getObject()  { return this.obj; }
    }
class Main
 {
   public static void main (String[] args)
   {
       Test <Integer> iObj = new Test<Integer>(15);
       System.out.println(iObj.getObject());
       Test <String> sObj =
                          new Test<String>("DataFlair");
       System.out.println(sObj.getObject());
   }
}
```

Java Generic Functions

We can also write generic functions in Java as they can be called by generic arguments which are handled by the compiler in Java.

```
class Test
 {
     static <T> void genericDisplay (T element)
        {
           System.out.println(element.getClass().getName() + " = " + element);
        }
     public static void main(String[] args)
          {
              genericDisplay(11);
              genericDisplay("datflair");
              genericDisplay(1.0);
          }
}
```

Advantages of Java Generics

Projects that utilizations Java Generics has numerous advantages over the non-generic code.

Advantages of Java Generics

Projects that utilizations Java Generics has numerous advantages over the non-generic code.

Advantages of Java Generics

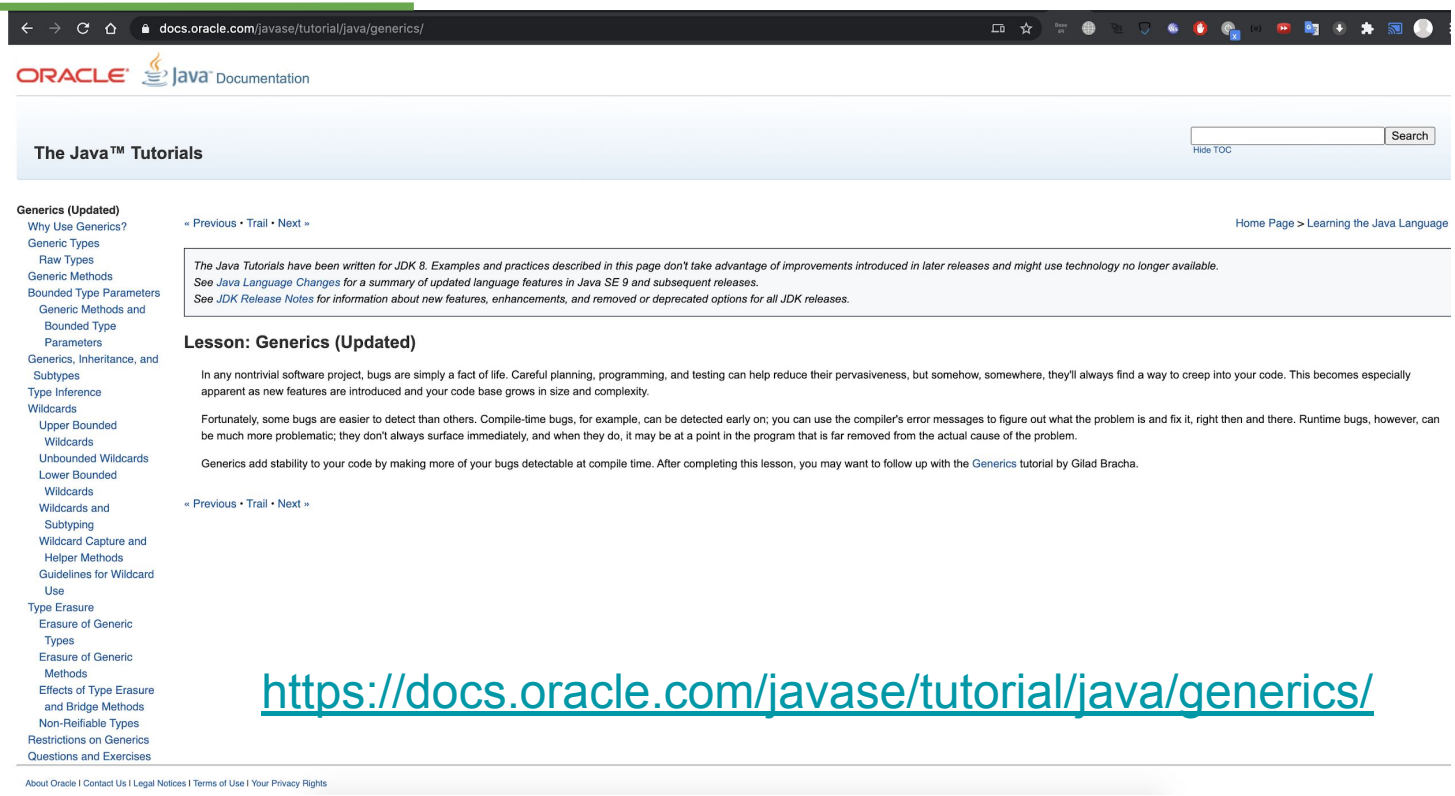
I. Code Reuse
II. Sort Safety
III. Individual Type Casting isn’t required
IV. Implementing non-generic algorithms

Advantages of Java Generics

I. Code Reuse
II. Sort Safety
III. Individual Type Casting isn't require
IV. Implementing non-generic algorith

Do your own research

For more details



https://docs.oracle.com/javase/tutorial/java/generics/

Warm up exercises

Warm up exercises

Will the following class compile? If not, why?

```
public final class Algorithm {
    public static <T> T max(T x, T y) {
        return x > y ? x : y;
    }
}
```

Warm up exercises

Will the following method compile? If not, why?

```
public static void print(List<? extends Number> list) {
    for (Number n : list)
        System.out.print(n + " ");
    System.out.println();
}
```

Warm up exercises

Will the following method compile? If not, why?

```
public class Singleton<T> {

    public static T getInstance() {
        if (instance == null)
            instance = new Singleton<T>();

        return instance;
    }

    private static T instance = null;
}
```

Warm up exercises

Given the following classes:

```
class Shape { /* ... */ }
class Circle extends Shape { /* ... */ }
class Rectangle extends Shape { /* ... */ }

class Node<T> { /* ... */ }
```

Will the following code compile? If not, why?

```
Node<Circle> nc = new Node<>();
Node<Shape>  ns = nc;
```

Warm up exercises

Consider this class:

```
class Node<T> implements Comparable<T> {
    public int compareTo(T obj) { /* ... */ }
    // ...
}
```

Will the following code compile? If not, why?

```
Node<String> node = new Node<>();
Comparable<String> comp = node;
```

Exercises...

Exercise 1

Design a class that acts as a library for the following kinds of media: book, video, and newspaper. Provide one version of the class that uses generics and one that does not. Feel free to use any additional APIs for storing and retrieving the media.

Exercise 2

Sketch the class definition and method signatures for a Stack behaviour (LIFO), parameterized by the type of element on the stack. Give the method signatures for push, pop, and isEmpty.

Exercise 3

Sketch the class definition and method signatures for a Dictionary class, which allows one to store or look up a value indexed by a key. Give the method signatures for get, put, isEmpty, keys, and values. The last two methods should return parameterized collections. (This class is similar to the builtin class HashMap in the Java collections library.)

References

- https://data-flair.training/blogs/java-generics/
- https://docs.oracle.com/javase/tutorial/java/generics/
- https://docs.oracle.com/javase/tutorial/java/generics/inheritance.html
- https://courses.cs.washington.edu/courses/cse341/08au/java/exercises.pdf