WIA1002/WIB1002 Data Structure

Tutorial: Generics

1. Create a generic class called Container that accepts one parameter, T. Create a no-arg constructor. Declare a private variable, t of type T. Create a method, add that returns nothing, accepting a parameter of generic type. Initialize this parameter to the initially declared variable. Create a generic method called retrieve() that returns the initially declared variable.

   Create a main method within the Container class. Create two containers of type Integer and String. Append two objects to the containers, one of type Integer having value 50, another of type String having value Java. Display the Integer and String objects from the respective containers.

```
public class Container<T> {
   private T t;

   // no-arg constructor
   public Container(){

   }

   public void add(T t){
      this.t = t;
   }

   public T retrieve(){
      return t;
   }

   public static void main(String[] args) {
      Container<Integer> c1 = new Container<>(); // auto-boxing since int is wrapped into Wrapper
Class Integer
      Container<String> c2 = new Container<>(); // no casting needed

      c1.add(50);
      c2.add("Java");

      System.out.println(c1.retrieve());
      System.out.println(c2.retrieve());
   }
}
```

2. Create a class called MyArray that has two methods, a main method that creates 3 arrays of
   a) integer containing the numbers 1,2,3,4 and 5
   b) string containing names, Jane, Tom and Bob
   c) character containing alphabet, a, b and c

   and a generic method listAll that displays the list of arrays.

```
public class MyArray {
   public static void main(String[] args) {
      Integer[] numbers = {1, 2, 3, 4, 5};
      String[] names = {"Jane", "Tom", "Bob"};
      Character[] alphabet = {'a', 'b', 'c'};

      listAll(numbers);
      listAll(names);
      listAll(alphabet);
   }

   public static <E> void listAll(E[] list){
      for (E elem: list) {
         System.out.print(elem + " ");
      }
      System.out.println("");
   }
}
```

3. What is a raw type? Why is a raw type unsafe? Why is the raw type allowed in Java?

   - Raw type is a generic class used without a type parameter.
   - A raw type is unsafe because the compiler loses the specific type information (datatype) that generics normally provide which can easily result in runtime errors.
   - The raw type is allowed in Java for backward compatibility with earlier versions of Java.

4. What is erasure? Why are Java generics implements using erasure?

   - Type erasure is an approach where the information on generics is used by the compiler but erases it afterward making the generic information is not available at runtime.
   - Java generics implements using erasure because it enables the generic code to be backward compatible with the legacy that uses raw types.

5. Create a generic class named Duo that has two parameters, A and B. Declare a variable named first of type A, and the second variable named second of type B. Create a constructor that accepts these two parameters. In the constructor, assign these parameters respectively to the declared variables.

```
public class Duo <A, B>{
   private A first;
   private B second;

   public Duo(A first, B second){ // Constructor
      this.first = first;
      this.second = second;
   }
}
```

6. Use the Duo class in Question 5 to declare and create two objects as follows :
   a. First object called sideShape consist of respectively String type and Integer type.
   b. Second object called points consists of two Double types.

```
public static void main(String[] args) {
      Duo<String, Integer> sideShape = new Duo<>("Square", 4);
      Duo<Double, Double> points = new Duo<>(0.1, 0.2);
   }
```

7. Assume that the following objects were created

```
ArrayList<String> vehicle = new ArrayList<>();
ArrayList<Object> transportation = new ArrayList<>();
```

Declare a method header for generic method,allTransportation that returns nothing,which accepts two ArrayList parameters using the wildcards.

```
public static <T> void allTransportation(ArrayList<T> t1, ArrayList<? super T> t2){}
```

8. Assuming that two new object are created as follows

```
ArrayList<Integer> numOfCars = new ArrayList<>();
ArrayList<Double> milesPerHour = new ArrayList<>();
```

Using the <?> wildcard, implement a generic method that displays the list.

```
public static void displayList(ArrayList<?> arr){
    for (Object elem: arr) {
        System.out.print(elem + " ");
    }
    System.out.println("");
}
```

9. When the compiler encounters a generic class, interface, or method with an unbound type parameter, such as <T> or <E>, it replaces all occurrences of the type parameter with what type?
   - Object type

10. When the compiler encounters a generic class, interface, or method with a bound type parameter, such as <T extends Number> or <E extends Comparable>, it replaces all occurrences of the type parameter with what type?
    - Bounded type

      e.g. Number, Comparable