LAB EXERCISE JAVA GENERIC CLASS

A generic type is a class or interface that is parameterized over types. We use angle brackets (<>) to specify the type parameter.

Please try all codes below

```
public class GenericsTypeOld {

	private Object t;

	public Object get() {
		return t;
	}

	public void set(Object t) {
		this.t = t;
	}

	   public static void main(String args[]){
	      GenericsTypeOld type = new GenericsTypeOld();
	      type.set("Java");
	      String str = (String) type.get();
	     //type casting, error prone and can cause ClassCastException
	}
}
```

Using Generic Class

```
public class GenericsType<T> {

	private T t;

	public T get(){
		return this.t;
	}

	public void set(T t1){
		this.t=t1;
	}

	public static void main(String args[]){
		GenericsType<String> type = new GenericsType<>();
		type.set("Java"); //valid

		GenericsType type1 = new GenericsType(); //raw type
		type1.set("Java"); //valid
		type1.set(10); //valid and autoboxing support
	}
}
```

Java Generic Interface

```
interface MinMax<T extends Comparable<T>> {
  T max(); /*  w  w w  .java2 s .  co m*/
}
class MyClass<T extends Comparable<T>> implements MinMax<T> {
  T[] vals;
  MyClass(T[] o) {
    vals = o;
  }
  public T max() {
    T v = vals[0];
    for (int i = 1; i < vals.length; i++) {
      if (vals[i].compareTo(v) > 0) {
        v = vals[i];
      }
    }
    return v;
  }
}

public class Main {
  public static void main(String args[]) {
    Integer inums[] = { 3, 6, 2, 8, 6 };
    Character chs[] = { 'b', 'r', 'p', 'w' };
    MyClass<Integer> a = new MyClass<Integer>(inums);
    MyClass<Character> b = new MyClass<Character>(chs);
    System.out.println(a.max());
    System.out.println(b.max());
  }
}
```

Java Generic Method

```
public class GenericsMethods {

    //Java Generic Method
    public static <T> boolean isEqual(GenericsType<T> g1, GenericsType<T> g2){
         return g1.get().equals(g2.get());
    }

    public static void main(String args[]){
         GenericsType<String> g1 = new GenericsType<>();
         g1.set("Java");

         GenericsType<String> g2 = new GenericsType<>();
         g2.set("Java");

         boolean isEqual = GenericsMethods.<String>isEqual(g1, g2);
         //above statement can be written simply as
         isEqual = GenericsMethods.isEqual(g1, g2);
```

```
         /*This feature, known as type inference, allows you to invoke
           a generic method as an ordinary method, without specifying a type
           between angle brackets */

         //Compiler will infer the type that is needed
      }
}
```

Java Generics Bounded Type Parameters

```
class Bound<T extends A>
{

    private T objRef;

    public Bound(T obj){
        this.objRef = obj;
    }

    public void doRunTest(){
        this.objRef.displayClass();
    }
}

class A
{
    public void displayClass()
    {
        System.out.println("Inside super class A");
    }
}

class B extends A
{
    public void displayClass()
    {
        System.out.println("Inside sub class B");
    }
}

class C extends A
{
    public void displayClass()
    {
        System.out.println("Inside sub class C");
    }
}

public class BoundedClass
{
    public static void main(String a[])
    {
```

```
        // Creating object of sub class C and
        // passing it to Bound as a type parameter.
        Bound<C> bec = new Bound<C>(new C());
        bec.doRunTest();

        // Creating object of sub class B and
        // passing it to Bound as a type parameter.
        Bound<B> beb = new Bound<B>(new B());
        beb.doRunTest();

        // similarly passing super class A
        Bound<A> bea = new Bound<A>(new A());
        bea.doRunTest();

    }
}
```

Java Generic WildCard

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;
import java.util.LinkedList;

/**
 * Wildcard Arguments With An Unknown Type
 * @author javaguides.net
 *
 */
public class WildCardSimpleExample {
 public static void printCollection(Collection<?> c) {
     for (Object e : c) {
         System.out.println(e);
     }
 }

 public static void main(String[] args) {
  Collection<String> collection = new ArrayList<>();
  collection.add("ArrayList Collection");
  printCollection(collection);
  Collection<String> collection2 = new LinkedList<>();
  collection2.add("LinkedList Collection");
  printCollection(collection2);
  Collection<String> collection3 = new HashSet<>();
  collection3.add("HashSet Collection");
  printCollection(collection3);

 }
}
```

Reference

- https://www.journaldev.com/1663/java-generics-example-method-class-interface#java-generics-class-subtyping
- www.java2s.com