Generics

- *Generics means Type parameter*
- *Generics introduces in java 5*
- *Generics detects Type parameter at compile time and avoid runtime error.*
- *While fetching the objects from Generics collection we don't required Type casting.*
- *Generics used at Compile time, at Runtime Generics information is removed.*

Type parameters

1. *T - Type*
2. *E - Element*
3. *K - Key*
4. *N - Number*
5. *V – Value*

Generics can be applied to Class, Interface, method, variable, return type.

Reference docs:

Lesson: Generics (Updated) (The Java™ Tutorials > Learning the Java Language) (oracle.com)

```
public class Generics1 {
    public static void main(String[] args) {

        List list = new ArrayList(); //without generics, before java-5
        list.add(1);//int -> Integer, Collections always store Wrapper types, not primitive Types, so 1 here
is Integer, int converted to Integer by Compiler. Compiler is smart :)
        list.add("two");
        list.add(3L);
        list.add(4.00);
        list.add(5.00F);

        Object i =   list.get(0);//everythign stored as an Object, to convert to Integer we need to cast ex-
plicitely
        Integer i2 = (Integer) list.get(1);//failed at runtime because "two" cannot be converted to Integer

    }
}
```

```
package com.hdfc.collections;

import java.util.ArrayList;
import java.util.List;

public class Generics1 {
    public static void main(String[] args) {

        List list = new ArrayList();
```

```
        list.add(1);//int -> Integer
        list.add("two");
        list.add(3);
        Integer i = (Integer) list.get(1);//failed at runtime
        Generics1.populateList(new ArrayList<>());
    }
    public static List<Integer> populateList(List<Integer> list){

        List<Object> list21 = new ArrayList<Object>();
        list21.add(1); //Integer
        list21.add("two");
        list21.add(3);
        list21.add("four");
        for(Object o: list21){

            //to print only String type
            /*if(o instanceof String){
                System.out.println(o);
            }*/

            //to print the type information
            System.out.println(o.getClass().getTypeName());
        }

        List list2 = new ArrayList();
        list2.add(1);
        list2.add(2);
        return list2;
    }

}
```

```
package com.hdfc.collections;


import java.util.ArrayList;
import java.util.List;

class Animal{}
class Dog extends Animal{}
public class Generics2 {
    public static void main(String[] args) {
        Animal a1= new Animal();
        Animal a2 = new Dog();
        Dog a3 = new Dog();
        //Dog a4 = new Animal();//not a valid

        //List<Animal> list  = new ArrayList<Dog>();//java: incom-
patible types: java.util.ArrayList<com.hdfc.collections.Dog> cannot
be converted to java.util.List<com.hdfc.collections.Animal>
```

```
        List<Animal> list  = new ArrayList<Animal>();
        list.add(new Animal());
        list.add(new Dog());

        //List<Animal> list2  = new ArrayList<Dog>(); //invalid
syntax

        List<Dog> list3  = new ArrayList<Dog>();
        //list3.add(new Animal());//cannot add Animal
        list3.add(new Dog());


        //List<Dog> list4  = new ArrayList<Animal>(); //invalid

        /*
        //Arrays
        Animal[] animals = new Animal[2];
        animals[0] = new Animal();
        animals[1] = new Dog();


        Animal[] animals2 = new Dog[2];
        animals2[0] = new Animal();
        animals2[1] = new Dog();

        Dog[] animals3 = new Dog[2];
        //animals3[0] = new Animal();
        animals3[1] = new Dog();

        //Dog[] animals4 = new Animal[2]; //in valid syntax
        */
    }
}
```

```
package com.hdfc.collections;


import java.util.ArrayList;
import java.util.List;


//NOTE: uncomment and see the error
class Vehical {
}

class Car extends Vehical {
}

class MarutiCar extends Car {
}

public class Generics3 {

    public static void main(String[] args) {
```

```
        List list = new ArrayList(); //can hold any object type , before java-5 syntax
        list.add(1);
        list.add(2L);
        list.add(3F);
        list.add(4D);
        list.add("hello");
        list.add(new Vehical());
        list.add(new Car());
        list.add(new MarutiCar());
        testBeforeJava5(list);//check the method signature
        testBeforeJava5_2(list);//check the method signature
        testBeforeJava5_3(list);//check the method signature
        testBeforeJava5_4(list);//check the method signature


        List<Object> list2 = new ArrayList<Object>(); //can hold any object type, after java-5 syntax
        testBeforeJava5(list2);//check the method signature
        testBeforeJava5_2(list2);//check the method signature
        testBeforeJava5_3(list2);//check the method signature
        testBeforeJava5_4(list2);//check the method signature


        List<Object> list3 = new ArrayList<>(); //can hold any object type, after java-7 syntax ArrayList<>();
need need to provide type information on right side

        List<?> list4 = new ArrayList<>(); //valid syntax but not allowed to store any car type
        testBeforeJava5(list4);//check the method signature
        //testBeforeJava5_2(list4);//check the method signature
        testBeforeJava5_3(list4);//check the method signature
        testBeforeJava5_4(list4);//check the method signature

        List<? extends Vehical> list5 = new ArrayList<>(); // Vehical and its subclasses
        List<? super Vehical> list6 = new ArrayList<>(); //Vehical and its super type

        //test1(List<? extends Vehical> list5)
        // test1(new )

        testAfterJava5(new ArrayList<Object>());
        testAfterJava5(new ArrayList<Integer>());
        testAfterJava5(new ArrayList<Vehical>());
        testAfterJava5(new ArrayList<Car>());
        testAfterJava5(new ArrayList<MarutiCar>());


        //testAfterJava5_1(List<? extends Vehical> list)
        //only those list which are extending Vehical are allowed
        // testAfterJava5_1(new ArrayList<Object>());
        //testAfterJava5_1(new ArrayList<Integer>());
        testAfterJava5_1(new ArrayList<Vehical>());
        testAfterJava5_1(new ArrayList<Car>());
        testAfterJava5_1(new ArrayList<MarutiCar>());

        //testAfterJava5_2(List<? extends Car> list){
        //only those list which are extending Car are allowed
        // testAfterJava5_1(new ArrayList<Object>());
        //testAfterJava5_1(new ArrayList<Integer>());
        //testAfterJava5_2(new ArrayList<Vehical>());
        testAfterJava5_2(new ArrayList<Car>());
        testAfterJava5_2(new ArrayList<MarutiCar>());


        //testAfterJava5_3(List<? extends MarutiCar> list){
        //only those list which are extending MarutiCar are allowed
        // testAfterJava5_3(new ArrayList<Object>());
        //testAfterJava5_3(new ArrayList<Integer>());
        //testAfterJava5_3(new ArrayList<Vehical>());
        //testAfterJava5_3(new ArrayList<Car>());
        testAfterJava5_3(new ArrayList<MarutiCar>());


        //testAfterJava5_4(List<? super Object> list){
        //only those list which are super type of Object are allowed
```

```
        testAfterJava5_4(new ArrayList<Object>()); //only objects are allowed, rest non
        //testAfterJava5_4(new ArrayList<Integer>());
        //testAfterJava5_4(new ArrayList<Vehical>());
        //testAfterJava5_4(new ArrayList<Car>());
        //testAfterJava5_4(new ArrayList<MarutiCar>());

        //testAfterJava5_5(List<? super Vehical> list){
        //only those list which are super type of Vehical are allowed
        testAfterJava5_5(new ArrayList<Object>());
        //testAfterJava5_5(new ArrayList<Integer>());
        testAfterJava5_5(new ArrayList<Vehical>());
        //testAfterJava5_5(new ArrayList<Car>());
        //testAfterJava5_5(new ArrayList<MarutiCar>());


        //testAfterJava5_6(List<? super Car> list){
        //only those list which are extending Car are allowed
        testAfterJava5_6(new ArrayList<Object>());
        //testAfterJava5_6(new ArrayList<Integer>());
        testAfterJava5_6(new ArrayList<Vehical>());
        testAfterJava5_6(new ArrayList<Car>());
        //testAfterJava5_6(new ArrayList<MarutiCar>());


    }

    public static void testBeforeJava5(List list) {

    }

    public static void testBeforeJava5_2(List<Object> list) {

    }

    public static void testBeforeJava5_3(List<?> list) {

    }

    public static void testBeforeJava5_4(List<? extends Object> list) {

    }

    public static void testAfterJava5(List<? extends Object> list) {

    }

    public static void testAfterJava5_1(List<? extends Vehical> list) {

    }

    public static void testAfterJava5_2(List<? extends Car> list) {

    }

    public static void testAfterJava5_3(List<? extends MarutiCar> list) {

    }


    public static void testAfterJava5_4(List<? super Object> list) {

    }

    public static void testAfterJava5_5(List<? super Vehical> list) {

    }

    public static void testAfterJava5_6(List<? super Car> list) {

    }

    public static void testAfterJava5_7(List<? super MyTest> list) {
```

 Contact: 8087883669

```
    }


}
```

```
package com.hdfc.collections;

class Test3 {
    int one;
    String two;
}

class UseTwo<T, X, Y> {
    T one;
    X two;
    Y three;
    public UseTwo(T one, X two, Y three) {
        this.one = one;
        this.two = two;
        this.three=three;
    }

    public T getT(){return this.one;}
    public X getX(){return this.two;}

    public Y getY(){return this.three;}

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

}

public class Generics4 {

    public static void main(String[] args) {

        UseTwo<Integer, Integer, Integer> object = new UseTwo<>(1, 1, 1);
        Integer t = object.getT();
        Integer x = object.getX();
        Integer y = object.getY();


        UseTwo<Integer, String, Double> object2 = new UseTwo<>(1, "two", 12.12D);
        Integer t1 = object2.getT();
        String x1 = object2.getX();
        Double y1 = object2.getY();

        UseTwo<Vehical, Car, MarutiCar> object3 = new UseTwo<>(new Vehical(), new Car(), new MarutiCar());
        Vehical t3 = object3.getT();
        Car x3 = object3.getX();
        MarutiCar y3 = object3.getY();


    }
}
```