Generics

Unit-4

• Generics, uses, advantages, bounded types

Generics:-

- Generics r a feat. of java wh. allows to create class, fun or interfaces in wh. the data type upon wh. they operate r specified as a parameter.
- Gen. r also called parameterized types.
- Suppose a class contains 2 var.
  In generics class, the same ~~two~~ 2 var can be ~~u~~ used as String, ~~I~~ Integer, Character etc.

Adv. of generics:-

1) Code reusability.
- Same class can be used for diff data type.
- We need not create separate classes for diff data types.

2) Type Safety :- **

- generics p/v type safety.
- obj of same class ~~to~~ but w/ diff ~~data~~ type parameters can't be assigned to each other
- this avoids runtime errors.

```
ArrayList<String> obj1 = new ArrayList<String>();
ArrayList<Character> obj2 = new ArrayList<Character>();
```

both obj r of class ArrayList
but they have diff type parameters - String & Character

So, they can't be assigned to each other.

Restrictions :-

- Generics only operate on obj not on primitive data types
- So wrapper ~~cat~~ classes Integer, Character should be used not int, char, etc.

ArrayList < Integer>
ArrayList < int > X

eg Create a gen class, it should contain 2 var. assign & print these variables in class.

```
class genclass <T> {
        ↳ ~~t~~ is type of parameter
          Can be any letter or more than 1 letter

    T obj1;
    T obj2;
    genclass (T po1, T po2){
        obj1 = po1;  obj2=po2;
    }
    public void get (){
        Sopln (obj1);
        Sopln(obj2);
    }
}

CD { PSVM () {

genclass<String> obj1 = new genclass <String>
                                ("abc", "def");
//obj1 contains el of type strings
obj1.get ();
//obj2 contains ~~two~~ el of type Integer.
~~obj~~ genclass<String> obj2 = new genclass <String>(10,20);
obj2.get ();

}
```

```
}
```

$\boxed{\text{Bounded Types}}$ ***.

- If you want the generic class to limit the data type in a generic class then it can be bounded by using extends Keyword.

- eg In the given eg:-
  obj of numeric types can only be created bcoz the type parameter T, extends Number class. wh. represents numerical data types.

```
~~genclass~~ Class  genclass <T [extends  Number]> {
                    T obj1;          ↳ only numeric data
                                       types allowed.
                    T obj2;
                    genclass (T po1, T po2) {
                            obj1 = po1;  obj2 = po2;
                    }

public void add() {
    double sum;
    sum = obj1.doubleValue()
          +obj2.doubleValue();   // convert to primitive
                                   data types bcoz
}                                  obj can't be added.

public void get() {
        Sopln(obj1);
        Sopln(obj2);
  }
}
```

```
class demo{
    PSVM(){

genclass <Integer> obj1 = new genclass <Integer>(10,20);
 obj1.add ;
      // Wrong!! string can't be used, bcoz it is bounded
genclass <String> obj2 = ----- ; X                  to
                                                Numeric
                                                 types.
                                                  only.
```

---

generic function :-

- fun can also be generic. Specify the type param b4 fun.
- gen fun can be created inside non generic class.

eg

```
CD{                       → gen fun.
 Public ~~void~~ static <T extends Number> void add (T a, T b){
             double sum = a.doubleValue() +
                              b.doubleValue();

             ~~sot~~ Sopln(sum);
  }
  PSVM(){
        add (10,20);

   }
  }
}
```