VIT-AP UNIVERSITY

Course code : **CSE2005**

Course title : **Object Oriented Programming**

Bounded Types

Objectives

This session will give the knowledge about

- Bounded Types
- Wildcard Arguments
- Bounded Wildcard

Bounded Types

Bounded types are used to limit the types that can be passed to a type parameter.

```
class Grade<Marks>{
        Marks[] obj;
        int sum;
        public Grade(Marks[] obj) {
                this.obj = obj; }
        public void findSum(){
                for(Object o:obj)
                        sum+=o.doubleValue();  //Error
        } }
```

Bounded Types

```
class Avg<Marks extends Number>{   //Here class Avg is bounded to Number
        Marks[] obj;
        int sum;
        public Avg(Marks[] obj) {
                this.obj = obj;
        }
        public double findAvg(){
                for(Number o:obj)
                        sum+=o.doubleValue();
                return sum/obj.length;
        }
}
```

Bounded Types

```
public class GenericDemo {
        public static void main(String[] args) {
                Integer iary[]={23,34,45,56};
                Avg<Integer> iobj=new Avg<Integer>(iary);
                System.out.println(iobj.findAvg());

                Double dary[]={53.45,34d,45.21,56d};
                Avg<Double> dobj=new Avg<Double>(dary);
                System.out.println(dobj.findAvg());

        }
}
```

Using Wildcard Arguments

Wildcard arguments means unknown type arguments. They just act as placeholder for real arguments to be passed while calling method.

They are denoted by question mark (?). One important thing is that the types which are used to declare wildcard arguments must be generic types. Wildcard arguments are declared in three ways.

- Wildcard Arguments With An Unknown Type
- Wildcard Arguments with An Upper Bound
- Wildcard Arguments with Lower Bound

Wildcard Arguments With An Unknown Type :

```
public class GenericDemo {
        static void processElements(ArrayList<?> a) {
                for (Object element : a) {
                        System.out.println(element);
                }
        }
        public static void main(String[] args) {
                ArrayList<Integer> a1 = new ArrayList<>();
                a1.add(10);
                a1.add(20);
                a1.add(30);
                processElements(a1);
```

Wildcard Arguments With An Unknown Type :

```
        ArrayList<String> a2 = new ArrayList<>();
        a2.add("One");
        a2.add("Two");
        a2.add("Three");
        processElements(a2);
    }
}
```

Wildcard Arguments With Upper Bound:

```
public class GenericDemo {
        static void processElements(ArrayList<? extends Number> a) {
                for (Object element : a) {
                        System.out.println(element);
                }
        }
        public static void main(String[] args) {
                ArrayList<Integer> a1 = new ArrayList<>();
                a1.add(10);
                a1.add(20);
                a1.add(30);
                processElements(a1);
```

Wildcard Arguments With Upper Bound :

```
        ArrayList<String> a2 = new ArrayList<>();
        a2.add("One");
        a2.add("Two");
        a2.add("Three");
        processElements(a2);
    }
}
```

Wildcard Arguments With Lower Bound:

```
public class GenericDemo {
        static void processElements(ArrayList<? super Integer> a) {
                for (Object element : a) {
                        System.out.println(element);
                }
        }
        public static void main(String[] args) {
                ArrayList<Integer> a1 = new ArrayList<>();
                a1.add(10);
                a1.add(20);
                a1.add(30);
                processElements(a1);
```

Wildcard Arguments

- Bounded and unbounded wildcards in generics are used to bound any Type.
- Type can be upper bounded by using <? extends T> where all Types must be sub-class of T, here T represent the upper bound
- Type can be lower bounded using <? super T> where all Types required to be the super class of T, here T represent the lower bound.
- Single <?> is called an unbounded wildcard in generic and it can represent any type, similar to Object in Java.

Bounded Wildcard :

```
class A{
        int x;
        A(int x){
                this.x=x;
        }
}
class B extends A{
        int y;
        B(int x,int y){
                super(x);
                this.y=y;
        }
}
```

Bounded Wildcard :

```
class C extends B{
        int z;
        C(int x,int y,int z){
                super(x,y);
                this.z=z;
        }
}
class Generics<T extends B>{
        T t;
        Generics(T t){
                this.t=t;
        }
}
```

Bounded Wildcard :

```
public class GenericDemo {
        public static void show1(Generics <?> g){
                System.out.println(g.t.x);
        }
        public static void show2(Generics <? extends B> g){
                System.out.println(g.t.x+" "+g.t.y);
        }
        public static void show3(Generics <? extends C> g){
                System.out.println(g.t.x+" "+g.t.y+" "+g.t.z);
        }
```

Bounded Wildcard :

```
        public static void main(String[] args) {
                A a=new A(12);
                B b=new B(12,23);
                C c=new C(12,23,34);

                Generics<B> g=new Generics<B>(b);
                show1(g);
                show2(g);
                show3(g); //Error
        }
}
```

Summary

We have discussed about

- Bounded Types
- Wildcard Arguments
- Bounded Wildcard