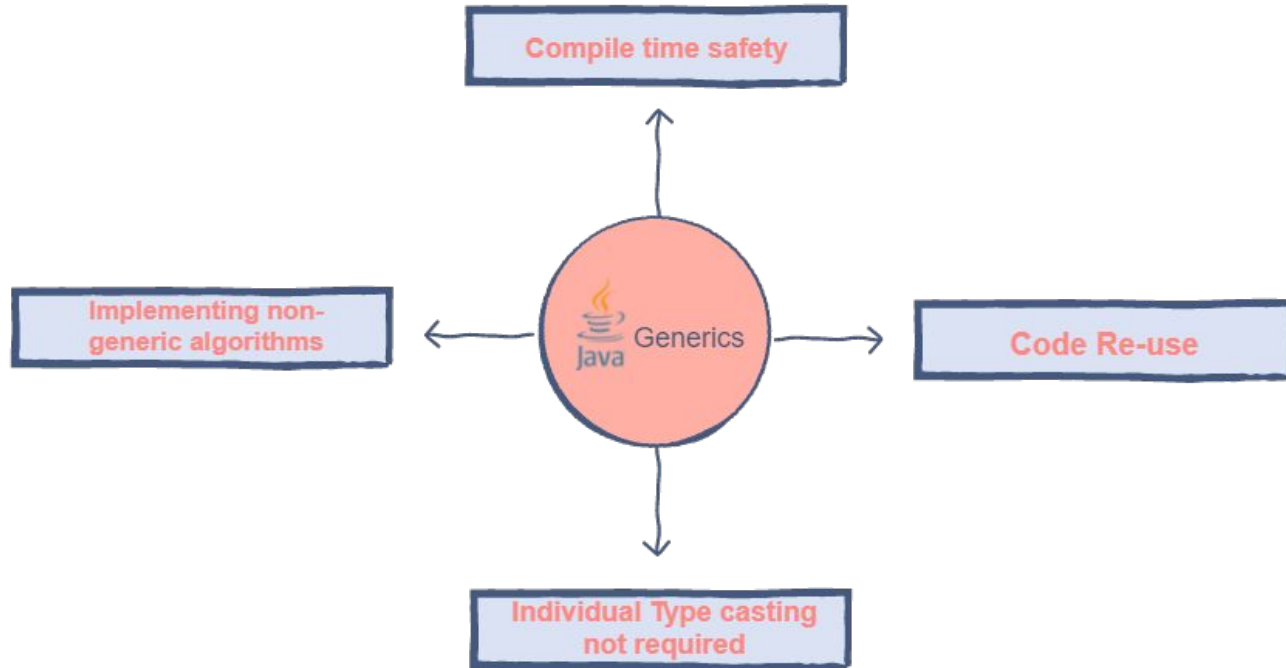


Generics

What are Generics?

At its core, the term generics means parameterized types. Parameterized types are important because they enable you to create classes, interfaces, and methods in which the type of data upon which they operate is specified as a parameter. Using generics, it is possible to create a single class, for example, that automatically works with different types of data. A class, interface, or method that operates on a parameterized type is called generic, as in generic class or generic method.

Type safety



Generic

```
class Sample<T>{  
  
}
```

Generic

```
class Sample<T> {  
    private Sample<String> sample;  
    private Sample<Integer> sample2;  
    //...  
}
```

General form

```
class class-name<type-param-list> { // ...
```

```
class-name<type-arg-list> var-name =  
    new class-name<type-arg-list>(cons-arg-list);
```

Any questions?



Bounded Types

O'rtacha qiymatni hisoblaydigan metod yozing.



Bounded Types

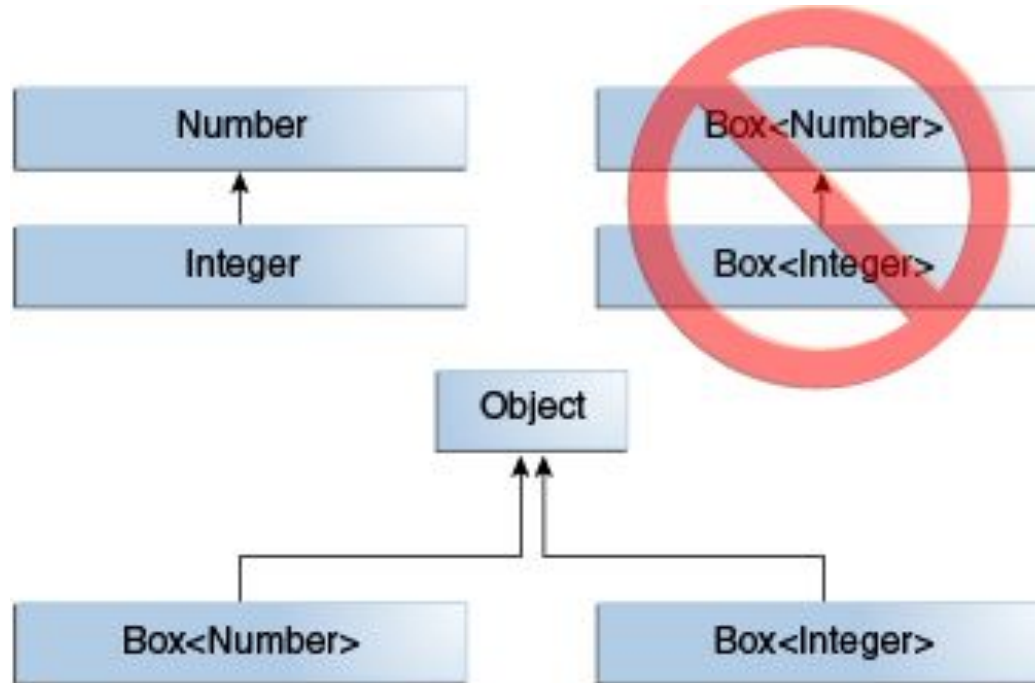
```
class Sample<T extends A> {
```

```
}
```

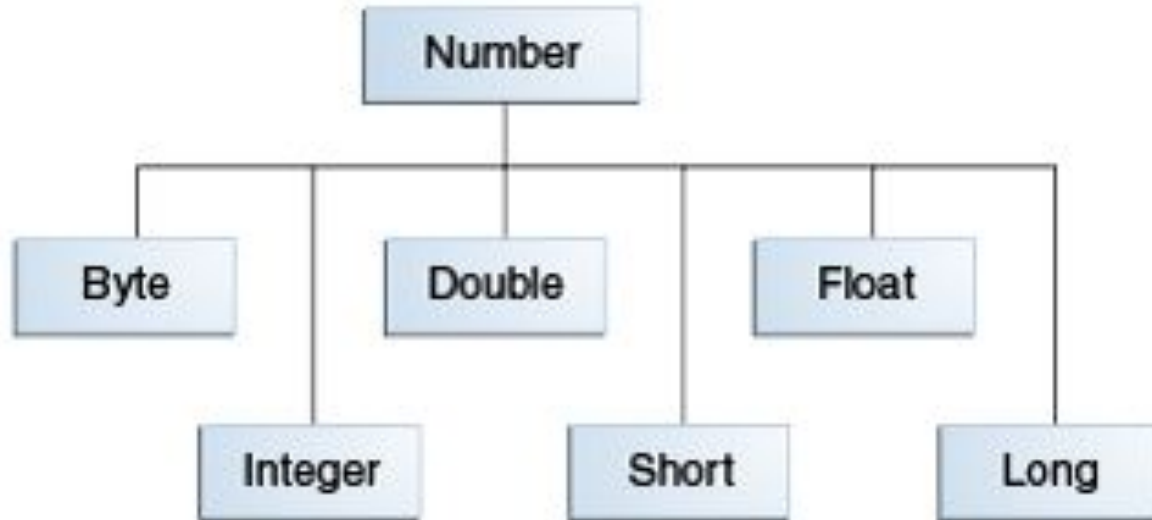
```
class A{
```

```
}
```










Bounded Types



Bounded Types



Number class methods

<p><<Java Class>></p> <p> Number</p> <p>java.lang</p>
<p> <u>serialVersionUID: long</u></p>
<p> Number()</p> <p> intValue():int</p> <p> longValue():long</p> <p> floatValue():float</p> <p> doubleValue():double</p> <p> byteValue():byte</p> <p> shortValue():short</p>

Bounded Types

```
public class Stats<T extends Number> {  
    private T[] nums;  
  
    public Stats(T... nums) {  
        this.nums = nums;  
    }  
  
    double average() {  
        double sum = 0.0;  
  
        for (T num : nums) {  
            sum += num.doubleValue();  
        }  
  
        return sum / nums.length;  
    }  
}
```

Any questions?



Wildcard

O'rtacha qiymatni solishtiradigan metod yozing.



Wildcard

```
boolean sameAverage(Stats<T> nums) {  
    return average() == nums.average();  
}
```


Wildcard

Yuqoridagi holatda qanday muammo bor ?



!Izoh: To'g'ri javob olgan birinchi o'quvchi 1 pointga erishadi.

Wildcard

```
boolean sameAverage(Stats<?> nums) {  
    return average() == nums.average();  
}
```

Wildcard

```
public static Sample<?> getInfo(Sample<?> a) {  
    return a;  
}
```

Any questions?



Bounded wildcard

`<? extends superclass>`

`<? super subclass>`

Any questions?



Generic method

```
<V> V showV(V obj) {  
    return obj;  
}
```

Generic method

```
<V extends Integer> int showV(V obj) {  
    return obj.intValue() + 10;  
}
```


Generic constructor

Qanday generic constructor yaratsak bo'ladi



Generic constructor

```
class GenCons {  
    private double val;  
  
    <T extends Number> GenCons(T arg) {  
        val = arg.doubleValue();  
    }  
  
    void showval() {  
        System.out.println("val: " + val);  
    }  
}
```

Any questions?



Generic Interfaces

```
interface Sample<T> {  
    void onVoid(T a);  
}
```

Generic Interfaces

```
interface Sample {  
    <T> void onVoid(T a);  
    <T> T onVoid2(T a);  
}
```

Generic Interfaces

```
interface Sample {  
    <T> void onVoid(T a);  
    <T> T onVoid2(T a);  
}  
  
class A implements Sample{  
  
    @Override  
    public <T> void onVoid(T a) {  
  
    }  
  
    @Override  
    public <T> T onVoid2(T a) {  
        return a;  
    }  
}
```

Any questions?



Generic class hierarchy

```
class A<T>{
```

```
}
```

```
class B<T,K> extends A<T>{
```

```
}
```