Wildcards - Introduction

```
class Glass<T>{ }
class Tray{
 public void add(Glass<Juice> juiceGlass){ }
}

class Tray{
 public void add(Glass<?> juiceGlass){ }
}
```

Unbounded Wildcards

```
class Tray{
 public void add(Glass<?> juiceGlass){ }
}
```

Unbounded Wildcard

Note: When the Generic class itself has Bounded Type parameters declared, We can only use Type arguments that are related to Bounded Types

When to use :

- When implementing the methods which should be able to work with any type.

Upper bounded Wildcards

```
class Glass<T>{ }
interface Liquid{ }
class Juice implements Liquid{ }
class OrangeJuice extends Juice{ }
```

```
class Tray{
    public void add(Glass<? extends Juice> juiceGlass){ }
}
```

Lower Bounded Wildcards

```
class Glass<T>{ }
class Juice{ }


class Coke{ }
class CokeDiet extends Coke{ }
class CokeZero extends CokeDiet{ }


class Tray{
 public void add(Glass<? extends Juice> juiceGlass){ }
 public void remove(Glass<? super CokeZero> colaGlass){ }
}
```

Upper Bound or Lower Bound

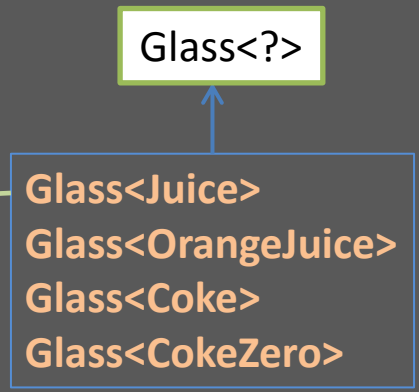
- Upper Bounded Wildcards
  - You keep the code open for extension to support any new types getting added to the type hierarchy.
- Lower Bounded Wildcards
  - You close the code to support any new types in the hierarchy and restrict to the types currently present .

Wildcards - Subtyping

```
Glass<Juice> juiceGlass =  new Glass<OrangeJuice>( );
```

```
class Tray{
 public void add(Glass<? extends Juice> juiceGlass){ }
 public void remove(Glass<? super CokeZero> colaGlass){ }
public void replace(Glass<?> juiceGlass){ }
}
```

```
public void replace(Glass<?> juiceGlass){ }
```



Wildcards - Subtyping

```
public void add(Glass<? extends Juice> juiceGlass) { }
```

```
interface Liquid{ }
class Juice implements Liquid{ }
class OrangeJuice implements Juice{ }
```

```
class Coke { }
class CokeDiet extends Coke{ }
class CokeZero extends CokeDiet{ }
```

Glass<Juice> ✓
Glass<OrangeJuice> ✓
Glass<Liquid> ✗

Glass<Coke>
Glass<CokeDiet>
Glass<CokeZero>

Wildcards Subtyping

```
public void remove(Glass<? super CokeZero> colaGlass){ }
```

```
class Coke { }
class CokeDiet extends Coke{ }
class CokeZero extends CokeDiet{ }
class CokeGreen extends Coke{ }
```

Glass<Coke> ✓

Glass<CokeZero> ✓

Glass<CokeGreen> ✗

Glass<Liquid> ✗

Glass<CokeDiet> ✓