

Bounded Type Parameters

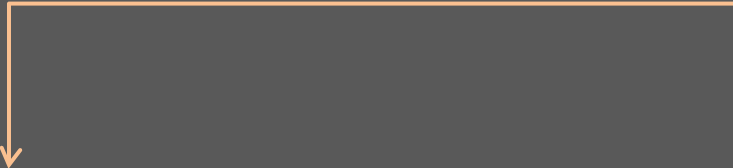
```
interface Liquid { }
```

```
class Glass<T>{  
    private T liquid;  
}
```

```
Glass<Cake> cakeGl = new Glass<Cake>( );
```

Bounded Type Parameter
extends keyword for both
Classes and Interfaces

```
class Glass<T extends Liquid>{  
    private T liquid;  
}
```

A diagram consisting of a horizontal line with a downward-pointing arrow at its end, connecting the 'extends' keyword in the class definition to the explanatory text.

Bounded Types - Instantiation

```
interface Liquid { }
```

```
class Juice implements Liquid{ }
```

```
class Water implements Liquid{ }
```

```
class Cake { }
```

```
class Diesel{ }
```

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

```
Glass<Water> waterGlass = new Glass<Water>( );
```

```
Glass<Cake> cakeGlass = new Glass<Cake>( );
```

```
Glass<Diesel> dieselGlass = new Glass<Diesel>( );
```



x

x

Bounded Types

```
interface Liquid {  
    public String taste( );  
}
```

```
class Juice implements Liquid{  
    public String taste(){  
        return "Sweet";  
    }  
}
```

```
class Water implements Liquid{  
    public String taste(){  
        return null;  
    }  
}
```

```
class Glass<T extends Liquid>{  
    private T liquid;
```

```
    public String getLiquidTaste(){  
        return liquid.taste();
```

```
    }  
}
```

You can always use the members of the Bounded Type mentioned in Type Parameters

Bounded Types - Methods

```
class Glass<T>{  
    private T liquid;  
  
    public String <U extends Juice> getLiquidTaste(U juice){  
        return juice.taste();  
    }  
}
```

Multiple Bounded Types

```
class Glass<T extends Juice & Fluid>{  
    private T liquid;  
}
```

```
class Juice{ }  
interface Fluid{ }
```

```
class OrangeJuice extends Juice implements Fluid{ }  
class AppleJuice extends Juice { }
```

```
Glass <OrangeJuice> orangeJuiceGlass = new Glass<OrangeJuice>( );  
Glass<AppleJuice> appleJuiceGlass = new Glass<AppleJuice>( );
```