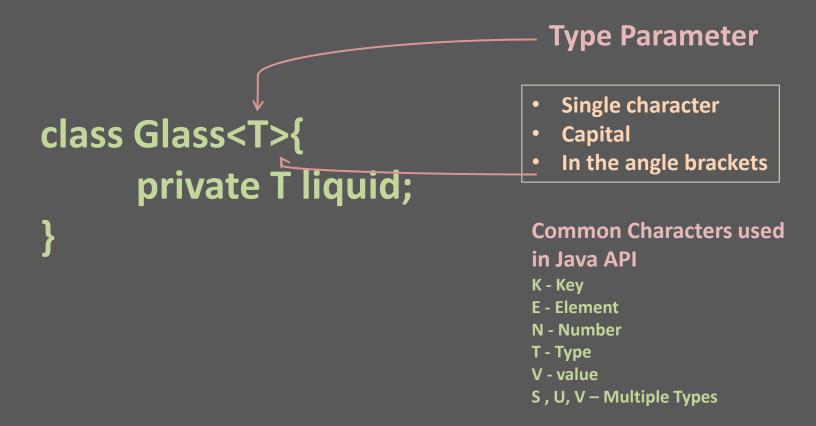
## **Generics Syntax**

**Classes and Interfaces** 



# Instantiating Generic Type

```
Type Parameter

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

```
Code using Generic Module
                         class Water { }
     class Juice{ }
Glass<Juice> juiceGlass = new Glass<Juice>();
Glass<Water> waterGlass= new Glass<Water>();
        Type Argument
Glass<Water> waterGlass= new Glass< >();
```

### Multiple Type Parameters

```
public class Colour<R,G,B> {
          public Colour(){ }
          public Colour(R red, G green, B blue){
          }
}
```

```
Red red = new Red();
Green green = new Green();
Blue blue = new Blue();
```

```
class Red { }
class Green{ }
class Blue{ }
```

```
Colour<Red, Green, Blue> colour = new Colour< >(red, green, blue);
```

```
Colour<Red, Green, Blue> colour = new Colour< >( );
```

#### **Generic Methods**

```
public class Bartender{
         //Method with return type
          public <J,W> Cocktail mix(J juice, W water){
         //Cocktail mixing code ;-)
          return new Cocktail();
          public <J,W> void mix2(J juice, W water){
          public static <J,W> void mix3(J juice, W water){
```

### **Generics Method - Invocation**

- No need to declare at class level to use at method level
- For variables Declare at class level

```
Bartender bartender = new Bartender();
Juice juice = new Juice();
Water water = new Water();
Glass<Cocktail> cocktail = bartender.<Juice, Water>mix(juice, water);
bartender.mix(juice, water); // Works - Type Inference
Bartender.<Juice, Water>mix3(juice, water); //Static method
```