Generic Classes

- **<T>**
- <T> takes any non-primitive object
- helps in avoiding type casting
- when we are extending Generic class to non-generic subclass, we need to define 'generic'
- we don't need to define 'generic' if we have generic sub-class
- We can make **one method as generic** too

```
public <T> returnType methodName () {
  }
```

- **Raw Type**
  - If we don't pass anything to generic, internally compiler will pass '**Object**' type to it
- **Bounded generic**
  - Bound types of objects generic takes
  - **Upper bound** - setting upper bound of object.
    - **<T extends Number>** accept all Number and its Child classes
  - **Multi bound** - extends one concrete class and implement interfaces
    - must be parent class or below it, and also implement interfaces
    - **<T extends ParentClass & interface1 & interface2>**
- **WildCards generic**
  - In wildcards we can provide different types, but in generics we need to have same type or defin second generic variable
  - In generic we don't have 'super' keyword
  - In 'generic types' we can have multiple generic type variables
  - **Upper bound wildcard** - same class and all its child classes are allowed
    - <? extends MainClass>
  - **Lower bound wildcard** - given class and above it
    - <? super SubClass>
  - **Unbounded wildcard** - when we don't know what type we can have
    - applies on 'Object'
    - <?>
- **Type Erasure**
  - <T> will be replaced with type when bytecode gets generated