Agenda

- Generic Limitations
- Generic Interfaces
  - Comparable
  - Comparator
- Arrays class
- ~~String Tokenizer~~
- ~~Garbage Collector~~

Generics Limitations

1. Cannot instantiate generic types with primitive Types. Only reference types are allowed.

```
ArrayList<Integer> list = new ArrayList<Integer>(); // okay
ArrayList<int> list = new ArrayList<int>(); // compiler error
```

2. Cannot create instances of Type parameters.

```
Integer i = new Integer(11); // okay
T obj = new T(); // error
```

3. Cannot declare static fields with generic type parameters.

```
   class Box<T> {
   private T obj; // okay
   private static T object; // compiler error
   // ...
   }
```

4. Cannot Use casts or instanceof with generic Type params.

```
if(obj instanceof T) {
newobj = (T)obj;
}
```

5. Cannot Create arrays of generic parameterized Types

```
T[] arr = new T[5]; // compiler error
```

6. Cannot create, catch, or throw Objects of Parameterized Types

```
throw new T(); // compiler error
try {
// ...
} catch(T ex) { // compiler error
// ...
}
```

7. Cannot overload a method just by changing generic type. Because after erasing/removing the type param, if params of two methods are same, then it is not allowed.

```
public void printBox(Box<Integer> b) {
// ...
}
public void printBox(Box<String> b) { // compiler error
// ...
}
```

Type erasure

- The generic type information is erased (not maintained) at runtime (in JVM). Box and Box both are internally (JVM level) treated as Box objects.
- The field "T obj" in Box class, is treated as "Object obj".
- Because of this method overloading with genric type difference is not allowed.

Generic Interfaces

- Interface is standard/specification.
- comparable is a predefined interface in java

```
// Comparable is pre-defined interface which was non-generic till Java 1.4

interface Comparable {
int compareTo(Object obj);
}

class Person implements Comparable {
// ...
public int compareTo(Object obj) {
Person other = (Person)obj; // down-casting
// compare "this" with "other" and return difference
}
}

class Program {
  public static void main(String[] args) {
    Person p1 = new Person("James Bond", 50);
    Person p2 = new Person("Ironman", 45);
```

```
    int diff = p1.compareTo(p2);
    if(diff == 0)
    System.out.println("Both are same");
    else if(diff > 0)
    System.out.println("p1 is greater than p2");
    else //if(diff < 0)
    System.out.println("p1 is less than p2");
    diff = p2.compareTo("Superman"); // will fail at runtime with
ClassCastException (in down-casting)
}
}
```

- Generic interface has type-safe methods (arguments and/or return-type).

```
// Comparable is pre-defined interface -- generic since Java 5.0
interface Comparable<T> {
int compareTo(T obj);
}

class Person implements Comparable<Person> {
// ...
public int compareTo(Person other) {
// compare "this" with "other" and return difference
}
}

class Program {
  public static void main(String[] args) {
    Person p1 = new Person("James Bond", 50);
    Person p2 = new Person("Ironman", 45);
    int diff = p1.compareTo(p2);
    if(diff == 0)
    System.out.println("Both are same");
    else if(diff > 0)
    System.out.println("p1 is greater than p2");
    else //if(diff < 0)
    System.out.println("p1 is less than p2");
    diff = p2.compareTo("Superman"); // compiler error
}
}
```


Comparable<>

- Standard for comparing the current object to the other object.
- Has single abstract method int compareTo(T other);
- In java.lang package.
- Used by various methods like Arrays.sort(Object[]), ...
- It does the comparision for the natural ordering

Comparator< >

- Standard for comparing two (other) objects.
- Has single abstract method int compare(T obj1, T obj2);
- In java.util package.
- Used by various methods like Arrays.sort(T[], comparator), ...

Arrays

- it is a class is java.util package
- cosists of helper methods for working on Array.
- eg
    - search
    - sort
    - equals
    - toString
- The array must be sorted (as by the sort() method) prior to making the search call.
- If it is not sorted, the results are undefined.

Assignment

Q. Create a student class with name,rollno,marks create a menu driven code that accepts the student and stores it inside the array. display all the students. display all students sorted on rollno in asc order display all students sorted on name in asc order display all students sorted on marks in desc order