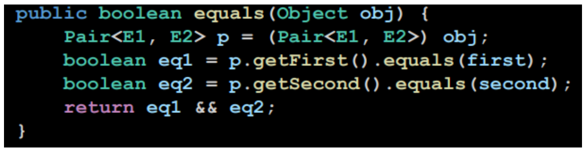
Comparing Multiple Generic Types



```
public boolean equals(Object obj) {
    Pair<E1, E2> p = (Pair<E1, E2>) obj;
    boolean eq1 = p.getFirst().equals(first);
    boolean eq2 = p.getSecond().equals(second);
    return eq1 && eq2;
}
```

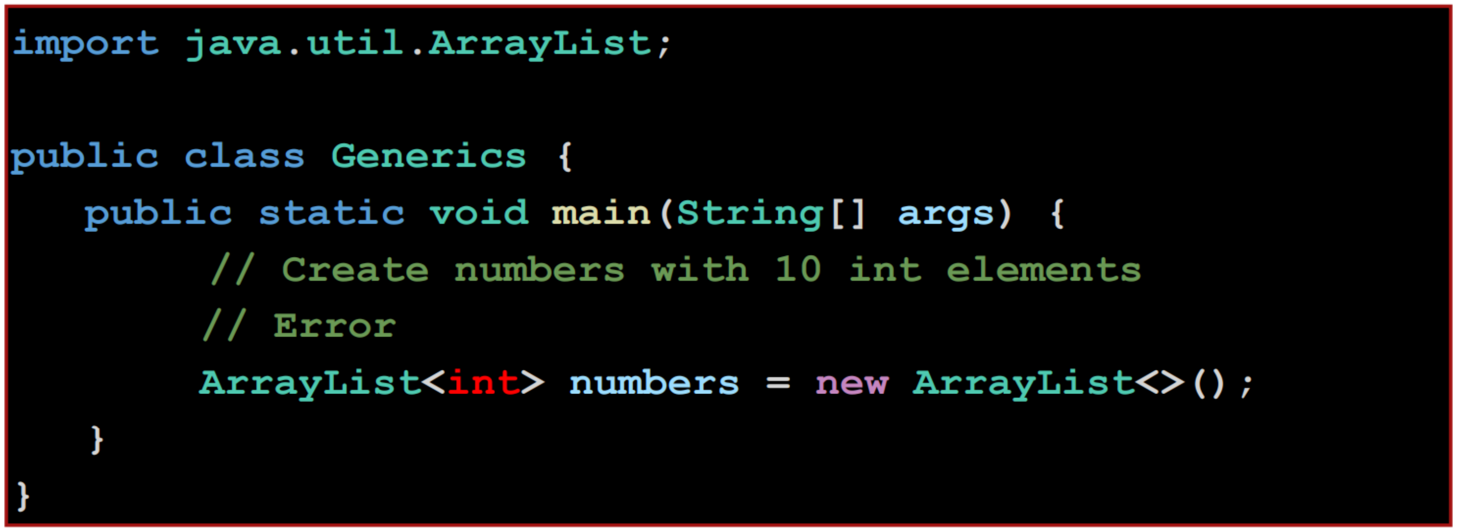
Generic Class Ex: `java.util.ArrayList;`

```
+ArrayList()
+ArrayList(int capacity)
+add(int index, E item): void
+add(E item): void
+get(int index): E
+set(int index, E item): E
+remove(int index): boolean
+size(): int
+isEmpty(): boolean
+clear(): void
+contains(Object obj): boolean
+indexOf(Object obj): int
+lastIndexOf(Object obj): int
+remove(Object obj): boolean
+remove(int index): boolean
```

- 2 Constructors:
  - no-arg creates an array of default size 10
  - One-arg creates an array of size `capacity`
- add (overloaded)
  - **`add(int index, E item)`**: adds `item` at location `index`.
    - All elements from `index` to `size()-1` are pushed one position up
  - **`add(E item)`**: adds `item` at first open location
- `get(int index)`: returns `item` at `index`
- `set(int index, E item)`: replaces element at location `index` with `item`
  - returns the old value of the `item` at `index`
- `remove(int index): boolean`
- `size()`: returns the actual size of the array (not `capacity`)
- `isEmpty()`: returns true if the array is empty
- `clear()`: reset size to 0
- `contains(Object obj)`: returns true if `obj` is in the array
- `indexOf(Object obj)`: returns the first index of `obj` if found, -1 otherwise
- `lastIndexOf(Object obj)`: returns the last index of `obj` if found, -1 otherwise
- Remove (overloaded):
  - `remove(Object obj)`: Returns true if `obj` is removed, and false otherwise
  - `remove(int index)`: Returns true if `index` is valid and element at `index` removed, false otherwise

```
Stack<E>
-elements: ArrayList<E>
+Stack()
+push(E item): void
+pop(): E
+peek(): E
+isEmpty(): boolean
+size(): int
+toString(): String
```

Common Error w/ `ArrayList`: Primitive Types



```
import java.util.ArrayList;

public class Generics {
   public static void main(String[] args) {
         // Create numbers with 10 int elements
        // Error
        ArrayList<int> numbers = new ArrayList<>();
   }
}
```

Restrictions on Generics

1. **Cannot create instances using the generic type `<E>`**
   a. The following is incorrect: `E item = new E();`
2. **Cannot create an array of type `E`**
   a. The following is incorrect: `E[] list = new E[20];`
3. **Generic type is not allowed in a static context**
   a. All instances of a generic class share same runtime class
   b. The following are incorrect:
   ```
   public static E item;
   public static void m(E object)
   ```
4. **Exceptions cannot be Generic**
   a. The following are incorrect:
   ```
   public class MyException<T> extends Exception{
   public static void main(String[] args){
     try{
        Cannot check the thrown exception
     }
     catch(MyException<T> ex){
     }
   }
   ```

Using Comparator to sort Shapes

```
<<Interface>>
java.util.Comparator;
int compare(T obj1, T obj2);
boolean equals(T obj);
```

```
java.util.Arrays;
<E> void sort(E[] list,Comparator<? Super E> c)
```

```
import java.util.Comparator;
public class ComparatorByColor implements Comparator<Shape> {
       public int compare(Shape s1, Shape s2){
      return s1.getColor().compareTo(s2.getColor());
   }
}
```

```
import java.util.Comparator;
public class ComparatorByArea implements Comparator<Shape>{
  public int compare(Shape s1, Shape s2){
      Double area1 = s1.getArea();
      Double area2 = s2.getArea();
      return area1.compareTo(area2);
    }
}
```

```
import java.util.Comparator;
public class ComparatorByColor implements Comparator<Shape> {
   public int compare(Shape s1, Shape s2){
      return s1.getColor().compareTo(s2.getColor());
   }
}
```

```
import java.util.Comparator;
public class ComparatorByArea implements Comparator<Shape>{
    public int compare(Shape s1, Shape s2){
        Double area1 = s1.getArea();
        Double area2 = s2.getArea();
        return area1.compareTo(area2);}}
```

```
public class TestShapeCmptr {
   public static void main(String[] args) {
      Shape[] s = { new Circle(),
                    new Circle("Red", 5.0),
                    new Circle("Blue", 2.5),
                    new Rectangle(),
                    new Rectangle("Green", 10.5, 5.5),
                    new Rectangle("Yellow", 4.0, 2.5) };
      printArray(s);
      System.out.println("\n");
      java.util.Arrays.sort(s, new ComparatorByArea());
      printArray(s);
      System.out.println("\n");
      java.util.Arrays.sort(s, new ComparatorByColor());
      printArray(s);
   }
```