Generic Classes and Methods

A generic method

```
class GenericMemoryCell
{
    public <AnyType> AnyType read(AnyType x)
    { return x;}
}
```

Its use

```
public static void main( String [] args)
  {

      GenericMemoryCell m = new GenericMemoryCell();
      Integer n = m.read(new Integer(7));
      System.out.println( "Contents are: " + n );
      String s = m.read("John");
      System.out.println( "Contents are: " + s );



  }
```

The findMax method again

```
class Auxiliaries {
public static MyComparable
    findMax( MyComparable [] arr) {
    int maxIndex = 0;

        for (int i = 1; i < arr.length; i++) {
        if( arr[i].compareTo( arr[ maxIndex ] ) > 0)
        maxIndex = i;
    }

        return arr[ maxIndex ];
  }
}
```

```java
interface MyComparable {
    public int compareTo(Object o);
}
```

```java
public static void main( String [] args) {
    IntHolder [] arr1 = { new IntHolder(11), new IntHolder(21),
                     new IntHolder(17), new IntHolder(34) };
    IntHolder ih = (IntHolder) Auxiliaries.findMax ( arr1);
    System.out.println(ih.i);

    Rectangl [] arr2 = {new Rectangl(5, 2), new Rectangl(5, 3),
            new Rectangl(6, 4),  new Rectangl(6, 2)};
    Rectangl rect = (Rectangl) Auxiliaries.findMax(arr2);
    System.out.println(rect.length + ", " + rect.breadth);
    }
```

```java
class IntHolder implements MyComparable {
    int i;
    public IntHolder (int i1) {
    i = i1;
    }
    public int compareTo(Object o) {
    IntHolder holder = (IntHolder) o;
        if (i > holder.i) return 1;
        if (i == holder.i) return 0;
        if (i < holder.i) return -1;
        return 0;
    }
}
```

```java
class Rectangl implements MyComparable {
    int length, breadth;
    public Rectangl (int l, int b) {
    length = l;
        breadth = b;
    }
    public int compareTo(Object o) {
        Rectangl rect = (Rectangl) o;
        if ( length * breadth > rect.length * rect.breadth ) return 1;
        if ( length * breadth == rect.length * rect.breadth ) return 0;
        if ( length * breadth < rect.length * rect.breadth ) return -1;
        return 0;
    }
}
```

The Issue

MyComparable [] arr3 = {new IntHolder(11), new Rectangl(5, 2)};
MyComparable comp = Auxiliaries.findMax(arr3);

Exception in thread "main" java.lang.ClassCastException:
IntHolder cannot be cast to Rectangl
        at Rectangl.compareTo(MyFindMax.java:60)
        at Auxiliaries.findMax(MyFindMax.java:12)
        at MyFindMax.main(MyFindMax.java:35)

A Generic interface

```
interface MyComparable<AnyType> {
    public int compareTo(AnyType o);
}
```

Class implementing interface

```
class IntHolder implements MyComparable<IntHolder> {
    int i;
    public IntHolder (int i1) {
    i = i1;
    }
    public int compareTo(IntHolder o) {
        if (i > o.i) return 1;
        if (i == o.i) return 0;
        if (i < o.i) return -1;
        return 0;
    }
}
```

The other class

```
class Rectangl implements MyComparable<Rectangl> {
    int length, breadth;
    public Rectangl (int l, int b) {
    length = l;
        breadth = b;
    }
    public int compareTo(Rectangl o) {
        if ( length * breadth > o.length * o.breadth ) return 1;
        if ( length * breadth == o.length * o.breadth ) return 0;
        if ( length * breadth < o.length * o.breadth ) return -1;
        return 0;
    }
}
```

Main

```java
IntHolder [] arr1 = { new IntHolder(11), new IntHolder(21),
                new IntHolder(17), new IntHolder(34) };
IntHolder ih = Auxiliaries.findMax ( arr1);
System.out.println(ih.i);

Rectangl [] arr2 = {new Rectangl(5, 2), new Rectangl(5, 3),
            new Rectangl(6, 4), new Rectangl(6, 2)};
Rectangl rect =  Auxiliaries.findMax(arr2);
System.out.println(rect.length + ", " + rect.breadth);
```

A generic findMax (won't compile)

```
class Auxiliaries {
public static <AnyType>
     AnyType findMax( AnyType [] arr) {
     int maxIndex = 0;

     for (int i = 1; i < arr.length; i++) {
     if( arr[i].compareTo( arr[ maxIndex ] ) > 0)
     maxIndex = i;
  }

     return arr[ maxIndex ];
  }
}
```

A generic findMax

```java
class Auxiliaries {
public static <AnyType extends MyComparable<AnyType>>
    AnyType findMax( AnyType [] arr) {
    int maxIndex = 0;

        for (int i = 1; i < arr.length; i++) {
        if( arr[i].compareTo( arr[ maxIndex ] ) > 0)
        maxIndex = i;
    }

        return arr[ maxIndex ];
    }
}
```

A Generic Class

```
class GenericMemoryCell<AnyType>
{

    public AnyType read()
    { return storedValue;}
    public void write( AnyType x) {
        storedValue = x; }
    private AnyType storedValue;
}
```

Using the Generic Class

```
class BoxingDemo {

    public static void main( String [] args)
    {
        GenericMemoryCell<Integer> m = new
            GenericMemoryCell<Integer> ();
        m.write(37);
        int val = m.read();
        System.out.println( "Contents are: " + val );
    }
}
```

Using the Generic Class

```
GenericMemoryCell<String> s = new
    GenericMemoryCell<String> ();
s.write("John");
String s1 = s.read();
System.out.println( "Contents are: " + s1 );
```

These won't compile

s.write(25);

m.write("Jack");

Our Old Stack

```
class MyStack {
    Object [] arr;
    int StackTop;
    public MyStack() { Code }

    public void push (Object o) { Code }

    public Object pop () { Code }

}
```

Its Use

```
MyStack sta = new MyStack();

sta.push("Jack");
sta.push(new Integer(7));
Object obj = sta.pop();
```

A Generic Stack Class

```
class MyStack<AnyType> {
    Object [] arr;
    int StackTop;
    public MyStack() { Code }

    public void push (AnyType o) { Code }

    public AnyType pop () { Code }
}
```

Its Use

```
MyStack<Integer> sta1 = new MyStack<Integer> ();
sta1.push( new Integer(17) );
sta1.push( new Integer(335));

Integer i = sta1.pop();

MyStack<String> sta2 = new MyStack<String> ();
sta2.push("Jack");
sta2.push("Jill");
String s = sta2.pop();
```

Code for class MyStack<AnyType>

```
Object [] arr;
int StackTop;

public MyStack() {
    arr = new Object [20];
    StackTop = -1;
}
```

code for class MyStack<AnyType>

```java
public void push (AnyType o) {
    StackTop++;
    arr[StackTop] = o;
}

public AnyType pop () {
    if (StackTop == -1) {return null;}
    else {
        StackTop--;
        return (AnyType) (arr[StackTop + 1]);
    }
}
```