CH. 2
OBJECT-ORIENTED PROGRAMMING

ACKNOWLEDGEMENT: THESE SLIDES ARE ADAPTED FROM SLIDES PROVIDED WITH DATA STRUCTURES AND ALGORITHMS IN JAVA, GOODRICH, TAMASSIA AND GOLDWASSER (WILEY 2016)
GENERIC PROGRAMMING

JAVA GENERICS
Generic Programming is a programming paradigm where the programmer programs interfaces and algorithms without a specific object hierarchy in mind. Rather the programmer only worries about operations (methods) needed to support an interface or perform an algorithm.
HOW DO WE PROGRAM GENERICALLY?

1. Convert things to raw memory, after all it is all 0's and 1's to the computer

2. Inheritance/polymorphism – Treat everything as an Object, or use very deep class hierarchies in combination with polymorphism

3. Generics/templates – A programming technique where we program without any specific type. Then when we instantiate a generic class/function the types becomes known

• (1) is outdated and for pure C programming (2) is done with polymorphism and (3) is done with Java Generics

• In all honesty though, we use (2) and (3) in combination.
SYNTAX FOR GENERIC OBJECTS

• Types can be declared using generic names:

1. `public class GenericClass<GenericType> {`
2. `private GenericType genericVariable;`
3. `/* Rest of class */`
4. `}`

• They are then instantiated using actual types:
  • `GenericClass<String> arr = new GenericClass<String>();`

• There is not much to it actually, but it is a very strange thought process when you do not know what E is as you write it.
**EXAMPLE**

**GENERIC ARRAY**

```java
public class Array<T> {
    private T data[];
    private int sz = 0;

    public Array(int cap) {
        data = new Object[cap];
    }

    public int size() { return sz; }
    public boolean isEmpty() { return sz == 0; }
    public T get(int i) { return data[i]; }
    public void add(int i, T elem) {
        if(sz == data.length) throw new IllegalStateException("Array is full");
        for(int k = sz; k > i; --k)
            data[k] = data[k - 1];
        data[i] = elem;
        ++sz;
    }

    /* For practice write the erase algorithm */
}
```

- They are then instantiated using actual types:
  - `Array<String> arr = new Array<String>();`
- They are used just like any other object:
  - `arr.add(0,"Hello");`
  - `String x = arr.get(0);`
GENERIC OBJECTS

• You can't call the constructor of the generic type, I'll explain more in a minute

• You may have one or more generic types. This class, we will have at most two
  • public class Map<Key, Value> {
  }  
  • Map<Integer, String>

• Generic types must be Java Objects, so you can use any class that inherits from Java Object, i.e., unfortunately you have to use Integer or Float instead of int or float.

• Many other quirks and oddities that will be experienced as we go!

• All code examples in the book and programming assignments involve this form of programming actually.
HOW IS IT ACCOMPLISHED IN JAVA?

• Java performs what is called **type erasure**

• Upon inserting an element into the data structure, Java stores it as a Java Object (i.e., genericly)

• When accessing an element, e.g., the get method from before, Java automatically inserts a type cast to preserve the type
You can also define Generic methods:

```
public static <T, S> String concat(T t, S s) {
    return t.toString() + s.toString();
}
```

Used like:

```
MyObject1 a;
MyObject2 b;
String c = concat(a, b);
```
EXERCISE

• Write a generic object `Pair`, which stores two things

• Write a generic method called `makePair` to easily construct new data pairs seamlessly

• Use this pair and generic method to create an array of random Cartesian coordinates from the unit square.

• Find the closest pair of points from this set and output it to the console.

• After I check your code, work on the programming assignment due Friday.