CS202: IT Workshop

Java

Arrays and ArrayList

Ref:

1. Harvey Deitel, Paul Deitel, **Java: How to Program**, 9/e, Prentice Hall India.

2. https://docs.oracle.com/

Last few lectures

- OOP concepts: Class, object, constructors
- **Inheritance**: programs with multiple classes
- **Polymorphism**: method overriding, overloading
- Abstract class, Interface
- Final members, static members

- **Any questions/suggestions/any info to share?**

Swapping program: pass by value

A Java program to swap two items

```
class Numbers {
        int item1, item2;

        Numbers (int i1, int i2) {
            item1 = i1; item2 = i2;
        }
        public void swap (int i1, int i2) {
            int temp = i1;
            i1= i2;
            i2 = temp;
        }
}
public class NumberDemo {
    public static void main(String args[])  {
        int num1 = 2;
        int num2 = 3;

        Numbers num = new Numbers (num1, num2);

        System.out.println("Items before swap: "+num.item1+" and "+num.item2);

        num.swap(num.item1, num.item2);

        System.out.println("Items after swap: "+num.item1+" and "+num.item2);
    }
}
```

Will the values be swapped?

Swapping program: pass by value

A Java program to swap two items

```
class Numbers {
        int item1, item2;

        Numbers (int i1, int i2) {
            item1 = i1; item2 = i2;
        }
        public void swap(int i1, int i2) {
            int temp = i1;
            i1= i2;
            i2 = temp;
        }
}
public class NumberDemo {
    public static void main(String args[])  {
        int num1 = 2;
        int num2 = 3;

        Numbers num = new Numbers (num1, num2);

        System.out.println("Items before swap: "+num.item1+" and "+num.item2);

        num.swap(num.item1, num.item2);

        System.out.println("Items after swap: "+num.item1+" and "+num.item2);
    }
}
```

```
$ Items before swap: 2 and 3

$ Items after swap: 2 and 3
```

No change: Swap method exchanged local variables only

Swapping program: operate on instance variable

A Java program to swap two items

```
class Numbers {
        int item1, item2;

        Number (int i1, int i2) {
            item1 = i1; item2 = i2;
        }
        public void swap() {
            int temp = item1;
            item1= item2;
            item2 = temp;
        }
}
public class NumberDemo {
    public static void main (String args[ ])  {
        int num1 = 2;
        int num2 = 3;

        Numbers num = new Numbers (num1, num2);

        System.out.println("Items before swap: "+num.item1+" and "+num.item2);

        num.swap();

        System.out.println("Items after swap: "+num.item1+" and "+num.item2);
    }
}
```

$ Items before swap: 2 and 3

$ Items after swap: 3 and 2

Value changed: Swap method exchanged instance variables of the object.

item1 and item2 are instance variables of class Numbers

Swapping program: passing object as parameter

A Java program to swap two items

```
class Numbers {
        int item1, item2;

        Number (int i1, int i2) {
            item1 = i1; item2 = i2;
        }
        public void swap(Number n) {
            int temp = n.item1;
            n.item1 = n.item2;
            n.item2 = temp;
        }
}
public class NumberDemo {
    public static void main(String args[])  {
        int num1 = 2;
        int num2 = 3;

        Numbers num = new Numbers (num1, num2);

        System.out.println("Items before swap: "+num.item1+" and "+num.item2);

        num.swap( num );

        System.out.println("Items after swap: "+num.item1+" and "+num.item2);
    }
}
```

```
$ Items before swap: 2 and 3

$ Items after swap: 3 and 2
```

n is an object; method is working on the object

As n is an object; Object is a reference type variable. Thus it is like **passed by reference.**
Hence changes are happening in the original place.

Arrays and ArrayList (Java support)

Using java.util.Arrays

- Working with array? Get help from Java's library
- The "Arrays" class of ***util*** package supports a number of static methods https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html

int [ ] *intArray* = { 7, 8, 1, 5, 9 };	$	Java implements dual-pivot quick sort in **sort()** method
Arrays.*sort(intArray);*	$ 1, 5, 7, 8, 9	
Arrays.*fill(intArray, 9);*	$ 9, 9, 9, 9, 9	Fills all the elements with 9
Arrays.*fill(intArray, 2, 4, 5);*	$ 9, 9, 5, 5, 9	Fills with 5 in the index range from 2 to 4-1
int loc = Arrays.*binarySearch(intArray, 5);*	$ 2	Returns the index of the first occurrence of the item 5

ArrayList in Java

- One of the limitation of array is its FIXED size
- ArrayList is a data structure where elements can be added, removed dynamically
- In addition, it supports various in-built methods
- It is a **generic** data structure under ***Collection*** framework of Java
- We can create an *ArrayList* of only **reference** types
- **What about primitive types (int, double etc.)?**
- ***Collection*** framework of Java supports other data structures like ***Set, List, Queue,*** etc.

ArrayList in Java

```
ArrayList< String > items = new ArrayList< String >();

items.add("red");
items.add(0, "yellow");
items.add("green");

for ( int i = 0; i < items.size(); i++ )
System.out.print( items.get( i ) );

items.remove(1);

for ( int i = 0; i < items.size(); i++ )
System.out.print( items.get( i ) );

System.out.print ( items.indexOf("green") );

items.clear();
```

```
$ yellow red green

$ yellow green

$ 1
```

add(item) inserts item at rear end

add (pos, item) inserts at the pos

remove(index) removes item given by index

get(index) returns item of index

indexOf(item) returns the index of the first occurrence of item

clear() removes all items

- We can create ArrayList of the Class we create in our code (e.g. Person)

Questions?

Wrapper class

- Many supported data structures in Java (such as Arraylist) works only with Reference type
- Wrapper class converts (wraps) primitive type to its equivalent Reference type (int → Integer, double → Double, etc.)
- Class "Integer" contains the equivalent int as a field within it (along with various other fields).

```
public final class Integer extends Number implements Comparable<Integer> {
        . . . . .
        private final int value;
        . . . . .
        public Integer (int value) {
                this.value = value;
                }
        . . . . .
```

Wrapper class example

- To create an integer object from a primitive int

```
Integer myIntObj = new Integer(5); //old version

Integer myIntObjNew = 7; //new version
```

This is an example of Auto boxing

- To get the equivalent int value,

```
int myPrimInt = myIntObj.intValue();
```

- Autoboxing and Unboxing is also supported

```
ArrayList<Integer> integerList = new ArrayList<Integer>();

integerList.add(24);

Character myCharObj = 'x';
char myChar = myCharObj;

System.out.println( myCharObj );
```

An example of unboxing

We can also print this way

Questions?

Command line argument

- We can pass information to an application as arguments (from command line)

```
$ javac CommandLineDemo.java

$ java CommandLineDemo 5 2 7
```

Every argument is passed as a String

We may need to convert them if required

- We accept these arguments in main() method and can process them

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

        int totArgument = args.length;

        int arrayLength = Integer.parseInt(args[0]);
        . . .
}
```

An array of Strings is received as argument

String is converted into an an integer