CST8130 – Data Structures

Professor : Dr. Anu Thomas
Email: thomasa@algonquincollege.com
Office: T314

Efficiency Concepts and related issues

Program Efficiency

Why does it matter?

How do you compare different programs?

- ***processing***
- ***memory***

How do you measure?

Memory

references – 32 bits (4 Bytes) locations

int – 32 bits (4 Bytes) – range - +- 2.1 billion

- byte – 8 bits (signed) – range is -128 to +127
- short – 16 bits (signed) – range is -32,768 to +32,767
- long – 64 bits (signed) – range is -9.223*10^{18} to +9.223*10^{18}

float – 32 bits (4 Bytes) – range -+/-1.4E-45 to +/- 3.4E+38;

8 decimal places of accuracy

double – 64 bits (8 Bytes) - Range is +/-4.94E-324 to

+/- 1.7977E+308; 17 decimal places of accuracy

char – 16 bits (2 Bytes) – unicode value

Memory implications/issues

- CHOOSE variables of correct size! Not too big (wasted memory) but also not too small (possible error with overflow) – depending on the data in your program

- watch where you declare variables. There is much overhead in ‘creating’ a variable in memory…

```
for (int i = 0;i < 1000; i++){
    int num = i*4;
    System.out.println (num);
}
```

```
int num;
for (int i = 0;i < 1000; i++){
    num = i*4;
    System.out.println (num);
}
```

```
for (int i = 0;i < 1000; i++){
    System.out.println (i*4);
}
```

Error conditions to think about

When reading in data

- Invalid – like chars when reading numeric
- Out of range – boundary checks

When allocating memory

- Number entries is not negative
- Enough memory is allocated off heap

When accessing array

- Invalid index– less than 0 or greater than length-1

Parameter Passing – example 1

```
Class Test {
  public static void main (String[]) {
         int num = 6;
         testMethod (num):  // this could be ANY method –
                            //  even one in another class
         System.out.println (“num is now” + num);
   }
   public static void testMethod (int num) {
          num = 5;
   }
}
```

This displays   num is now 6

Parameter Passing – example 1

- Declared a primitive variable
- Passed it as parameter to a method
- Changed value of parameter value in method
- Printed variable when method has finished back in calling method – has not changed

Parameter Passing – example 2

```
Class Test {
  public static void main (String[]) {
        int num = 6;
        num = testMethod (num);
        System.out.println (“num is now” + num);
   }
   public static int testMethod (int num) {
         return (num -1);
   }
}
```

This displays num is now 5

Parameter Passing – example 2

- Declared a primitive variable
- Passed it as parameter to a method
- Changed value of parameter value in method
- Returned the updated value from the method and stored it back into original variable
- Printed variable when method has finished back in calling method – has new value

Parameter Passing – example 3

```
class Data {
    private int num;
    public Data () { num = 6; }
    public int testMethod (int num) {return (num -1);}
    public String toString() { return “in Data num is “ + num}
}
In main…..   Data dataObj = new Data();
             System.out.println (dataObj); // in Data num is 6
             System.out.println (dataObj.testMethod(4));  // 3
        System.out.println (dataObj); // in Data num is 6
```

Parameter Passing – example 3

- Declared a primitive field in a class
- In main, created an object which initialized the primitive field value
- Called toString to display the value of the field – it has the initial value

Parameter Passing – example 4

```
class Data {
    private int []num;
    public Data () { num = new int[3];}
    public void testMethod (int x) { num = new int[x]; }
    public String toString() { return “in Data len is “ + num.length;  }
}
In main…..   Data dataObj = new Data();
             System.out.println (dataObj); // in Data len is 3
       dataObj.testMethod(23);
       System.out.println (dataObj); // in Data len is 23
```

Parameter Passing – example 4

- Declared a reference field in a class (an array)
- In main, created an object which initialized the reference field value (created an array of length 3)
- Called toString to display the value of the field – it has the initial value – array of length 3
- Called a method on the object that used the parameter to change the length of array to one of that parameter length – so changed the field value
- Called toString to display the value of the field – and the changed value is displayed

Parameter Passing – example 5

```
Class Data {
        private int num;
        public Data (int x) { num = x; }
        public String toString() { return “in Data num is “ + num;  }
}
Class DataTest {
        private Data dataObj = new Data(12);
        public String toString( ){return “ Data Test ” + dataObj.toString();}
}
In main…..   DataTest   obj = new DataTest();
                  Data mainObj = new Data(43);
                         System.out.println (obj); // Data Test in Data num is 12
                  System.out.println (mainObj); // in Data num is 43
```

Parameter Passing – example 5

- Created a class (Data) with primitive field (num) and another class (DataTest) which has reference field – an object of Data class – which in turns has the primitive field (num)
- In main – created a DataTest object – which initializes the Data object field to 12
- In main – created a Data object – initialized Data object field to 43
- Called toString on DataTest object to verify the field (num) is 12
- Called toString on Data object to verify the field (num) is 43

Parameter Passing – example 5 contd.

```
Class Data {
        private int num;
        public Data (int x) { num = x; }
        public String toString() { return “in Data num is “ + num;  }
}
Class DataTest {
        private Data dataObj = new Data(12);
        public void update (Data newData) { dataObj = newData; }
        public String toString( ){return “ Data Test ” + dataObj.toString();}
}
In main…..   DataTest  obj = new DataTest();
                Data mainObj = new Data(43);
                obj.update (mainObj);
                System.out.println (obj); // Data Test in Data num is 43
```

Parameter Passing – example 5 contd.

- Wrote a method in DataTest class to update the object of Data field to parameter passed in …note that this method uses the equal operator which just copies reference values
- Back in main, we called this update method on our DataTest object (which has num of 12) – using the object of Data (which has num of 43) as parameter
- Called toString on the DataTest object – and we see that it now has num of 43

Parameter Passing – example 5 cont2

```
Class Data {
        private int num;
        public Data (int x) { num = x; }
             public void update (int x) { num = x; }
        public String toString() { return “in Data num is “ + num;  }
}
Class DataTest {
        private Data dataObj = new Data(12);
        public void update (Data newData) { dataObj = newData;
                                              newData = new Data(34);}
        public String toString( ){return “ Data Test ” + dataObj.toString();}
}
In main…..      DataTest  obj = new DataTest();
                Data mainObj = new Data(43);
                obj.update (mainObj);
                mainObj.update (22);
                System.out.println (obj); // Data Test in Data num is 22
                System.out.println (mainObj); // in Data num is  22
```

Parameter Passing – example 5 cont2

- Now we update the num in the object of type Data in our main so it has a value of 22 by calling a method to update num
- When we print the object of DataTest again – it now has the value of 22 in num (because it is still pointing to the previous memory belonging to the object of Data)

Questions?