Variables and References

- Java has two type of variables
  - Value variables
    - Names being with lower case letter
      - int, double, ...
  - Reference Variables
    - Names begin with upper Case Letters
      - String, and all classes

Variable initialization

- Value variables
  - Declare the name == allocate space and set a value
    - int aa;
      - creates variable aa with value 0
- Reference Variables
  - Declare a name makes a name and gives it a value of **`null`**
    - String aa;
  - “new” keyword allocates space and returns the location of the allocated space
    - aa = new String(“bb”);

By_ref and By-value Variables

```
public class T1 {
    public static void incII(MyI i3) {
       i3.setII(1+i3.getII());
       System.out.println("ii="+i3);
    }
    public static void incI(int i4) {
       i4 = i4 + 1;
       System.out.println("i="+i4);
    }
    public static void main(String[] args)
    {
       MyI i1 = new MyI();
       int i2 = 0;
       System.out.println(i1 + " " + i2);
       incII(i1);
       incI(i2);
       System.out.println(i1 + " " + i2);
    }}
class MyI {
    private int ii=0;
    public int getII() { return ii; }
    public void setII(int ii) { this.ii=ii;}
    public String toString() { return ""+ii; }
}
```

Classes and Inheritance

- “extends” means that one class inherits from another
  - inherited things
    - public and protected methods
    - public and protected variables
  - NOT inherited things
    - private variables
    - private methods
    - overwritten methods

Overwriting and Overloading

- Methods of a class may be “overloaded”
  - same name,
  - same return value,
  - different arguements
- inherited methods may be overwritten
  - same name
  - same return value
  - same arguements
  - Applets: paint, init, ...
  - Applications: main, toString, ...
- NOTE
  - same name requires same return value

O & O

```
public class T2
{
   public String toString(int i) { return "aaaa" + i; }
   public String toString(int i,String s) {return s+i;}
}

class T3 extends T2
{
   public String toString() { return "bbbb"; }
   public static void main(String[] args) {
     T2 i2 = new T2();
     T3 i3 = new T3();
     System.out.println(i2.toString());
     System.out.println(i3.toString());
     System.out.println(i2.toString(5));
     System.out.println(i3.toString(5));
     System.out.println(i2.toString(5, "a"));
     System.out.println(i3.toString(5, "a"));
   }
}
```

Order Estimates

- Make time (or space) estimates based on some quantity – typically input
- Ignore constant factors
- Typically from a small set of functions
- Worry about what happens when n is large
  - O(n), O(n^2), O(n^3), O(2^n), O(1), O(log log n), O(n * log n), O (log n), O(2^n^n)

Searching

- Linear
  - O(n)
  - works on everything
- Binary
  - O(log n)
  - requires items be sorted
  - requires items are in a data structure allowing non-sequential access

Sorting

- O(n^2) for all algorithms
- Bubble
  - compare neighbors, if not in order, swap
- Selection
  - find the smallest, put it first. Find the second smallest, put it second, ...
- Insertion
  - assume the first items is sorted. Move the second item so that the first two are sorted. Move the third item ...

Stacks

- Last in – first out
- Basic operations
  - push
  - pop
  - size
- If stack implemented using arrays need a system for increasing the size of the stack

Queues

- first in – first out
- Basic Operations
  - enqueue
  - dequeue
  - size
- If implemented using arrays need a system for increasing the size

Using a queue to make a stack

- Problem:
  - I have a queue implementation.
  - I want a stack.
  - I want to extend the Queue to make a stack
  - I want all operations to be O(n).
  - I want to use only 1 queue
- Question: Can I do this? How?

Using a stack to make a queue

- Problem:
  - I have a stack implementation.
  - I want a queue.
  - I want to extend the stack to make a queue
  - I want all operations to be O(n).
  - I want to use only 1 stack
- Question: Can I do this? How?

Linked Lists

- Typically implemented using two classes
- Node
  - holds data items
  - holds next pointer
- Linked List
  - holds pointer to start of list
  - convenience methods