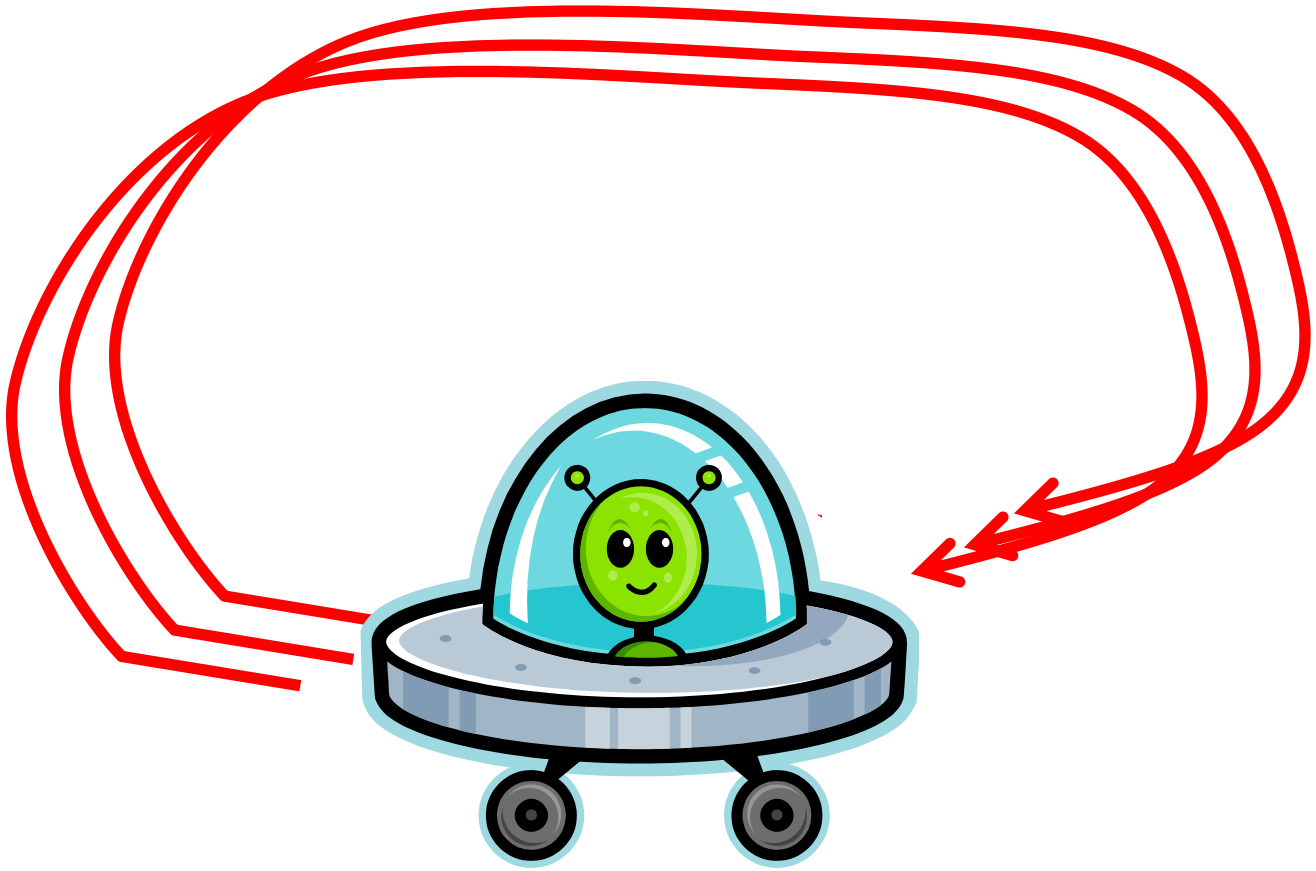
More on objects and looping



Overview

- Default values
  - Local variables, instance variables, arrays
- Object oriented programming
  - Multiple files, multiple main() methods
- More on loops
  - Enhanced loops
  - Breaking out early

Default values

- Local variables
  - Must be initialized before use
- Instance variables
  - Get a default value
- Elements of an array
  - Get a default value

type	default value
int	0
double	0.0
boolean	false
String	null
reference type	null

Java compiler will complain:
`The local variable x may not have been initialized`

```
public static void main(String [] args)
{
    int x;
    for (int y = 0; y < 5; y++)
    {
        x = x + y;
    }
}
```

How smart is the compiler?

- Not very…

Java compiler will still complain:
`The local variable x may not have been initialized`

We know `Math.random()` is always in [0.0, 1.0) which is always < 1.0. So `x` will always be set, but Java can't figure that out!

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

   if (Math.random() < 1.0)
      x = 0;
   for (int y = 0; y < 5; y++)
   {
      x = x + y;
   }
}
```

Instance variable default values

- Instance variables get a default value
  - Remember: reference variables default to null!
  - Must create with new before you use
    - When you declare the instance variable
    - In the constructor

```
public class Ball
{
   private double posX   = 0.0;
   private double posY   = 0.0;
   private double radius = 0.0;
   private Color  color  = new Color(0.88f, 0.68f, 1.0f);

   ...
}
```

Java would set these as the default values, (but doesn't hurt to be explicit).

Java would set this to `null` by default, but we create a nice default color

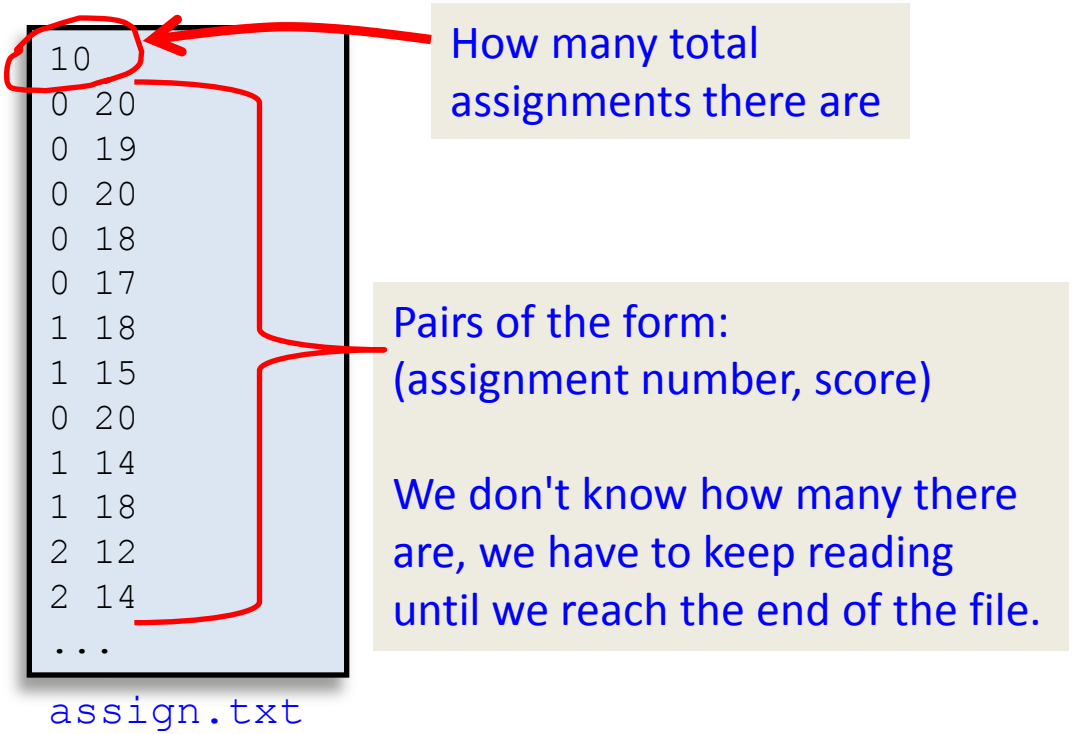
Instance variable default values

```
public class Square
{
   private double posX;
   private double posY;
   private double size;
   private Color  color;

   public String toString()
   {
      String result = "(" + posX + ", " + posY + ") ";
      result += "size = " + size;
      result += "color = (";
      result += color.getRed() + " ";
      result += color.getGreen() + " ";
      result += color.getBlue() + ") ";
      return result;
   }

   public static void main(String [] args)
   {
      Square s = new Square();
      System.out.println(s);
   }
}
```

Unless some other method creates the Color object, a call to toString() will runtime error:
`java.lang.NullPointerException`

Array default values

- Elements get default for type array holds
  - If you want something else → do it yourself
- Example:
  - Goal: find min score on a set of assignments



Minimum score example

```
public static void main(String [] args)
{
   // Read in how many total assignments there are
   int num = ???1???

   // Declare and create our array that tracks the min score
   ???2??? minScore = ???3???

   // Change the default value stored in the array's elements
   for (int i = ???4???)
       ???5???

   // Loop until we run out of input in the file
   while (???6???)
   {
      // Read in the next pair
      int assignment = ???7???
      int score = ???8???

      // Check if this is a new low for the assignment
      if (score < minScore[assignment])
         ???9???
   }
   // Print out the results
   for (int i = 0; i < minScore.length; i++)
      System.out.println("Assignment " + i + ": " + minScore[i]);
}
```

```
10
0 20
0 19
0 20
0 18
0 17
1 18
1 15
0 20
1 14
1 18
2 12
2 14
...
```

assign.txt

Minimum score solution

```
public static void main(String [] args)
{
   // Read in how many total assignments there are
   int num = StdIn.readInt();

   // Declare and create our array that tracks the min score
   int [] minScore = new int[num];

   // Change the default value stored in the array's elements
   for (int i = 0; i < minScore.length; i++)
       minScore[i] = Integer.MAX_VALUE;

   // Loop until we run out of input in the file
   while (!StdIn.isEmpty())
   {
      // Read in the next pair
      int assignment = StdIn.readInt();
      int score = StdIn.readInt();

      // Check if this is a new low for the assignment
      if (score < minScore[assignment])
         minScore[assignment] = score;
   }
   // Print out the results
   for (int i = 0; i < minScore.length; i++)
      System.out.println("Assignment " + i + ": " + minScore[i]);
}
```

```
10
0 20
0 19
0 20
0 18
0 17
1 18
1 15
0 20
1 14
1 18
2 12
2 14
...
```

assign.txt

OOP Quiz

- Classes and objects

Class =

Object =

Instance variables =

Instance methods =

Constructor =

OPP Quiz

- Classes and objects

Class = object blueprint

Object = instance of a class

Instance variables = what an object knows

Instance methods = what an object can do

Constructor = object stork, creates an instance from the blueprint

Building software

```
public class MyProgram
{
   public static void
   main(String [] args)
   {
     // Lots and lots of stuff
     // goes here.  All the
     // logic and state of my
     // software solution.
   }
}
```

MyProgram.java

How we use to do it (before objects)

Our software is now a collection of files. A main program and a bunch of objects cooperating to Get the Job Done™.

```
public class MyProgram
{
   public static void
   main(String [] args)
   {
     // Some stuff still goes here,
     // like creating objects help us
     // implement our software solution
   }
}
```

MyProgram.java

```
public class MyObj1
{
   // Instance variables
   // Instance methods
   ...
}
```

MyObj1.java

```
public class MyObj2
{
   // Instance variables
   // Instance methods
   ...
}
```

MyObj2.java

Multiple `main()` methods

- Every class can have a `main` method
  - Uses the one in the *.class file given to `java`

```
public class Ball
{
   private double posX = 0.0;
   ...
   public static void
   main(String [] args)
   {
      Ball a = new Ball(0.5, 0.5, 0.2);
      System.out.println("a = " + a);
      a.move(0.5, 0.5);
      System.out.println("now = " + a);
   }
}
```

Often class designers will create a `main()` method that is used for testing out the methods in the class.

```
public class BallGoingUp
{
   public static void
   main(String [] args)
   {
      Ball b = new Ball(0.5, 0.5, 0.1);
      while (true)
      {
        b.move(0.0, 0.01);
        b.draw()
        StdDraw.show(100);
      }
   }
}
```

One possible client program that makes use of the `Ball` class. What will this draw?

Multiple `main()` methods

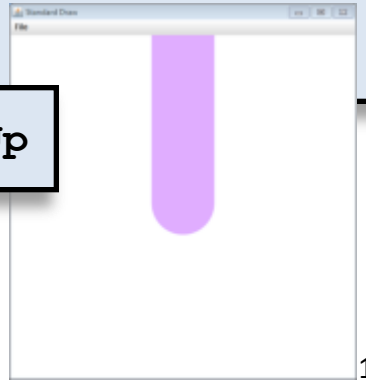
- Every class can have a `main` method
  - Uses the one in the *.class file given to `java`

```
public class Ball
{
   private double posX = 0.0;
   ...
   public static void
   main(String [] args)
   {
      Ball a = new Ball(0.5, 0.5, 0.2);
      System.out.println("a = " + a);
      a.move(0.5, 0.5);
      System.out.println("now = " + a);
   }
}
```

```
% java Ball
a = (0.5, 0.5) r = 0.2
now = (1.0, 1.0) r = 0.2
```

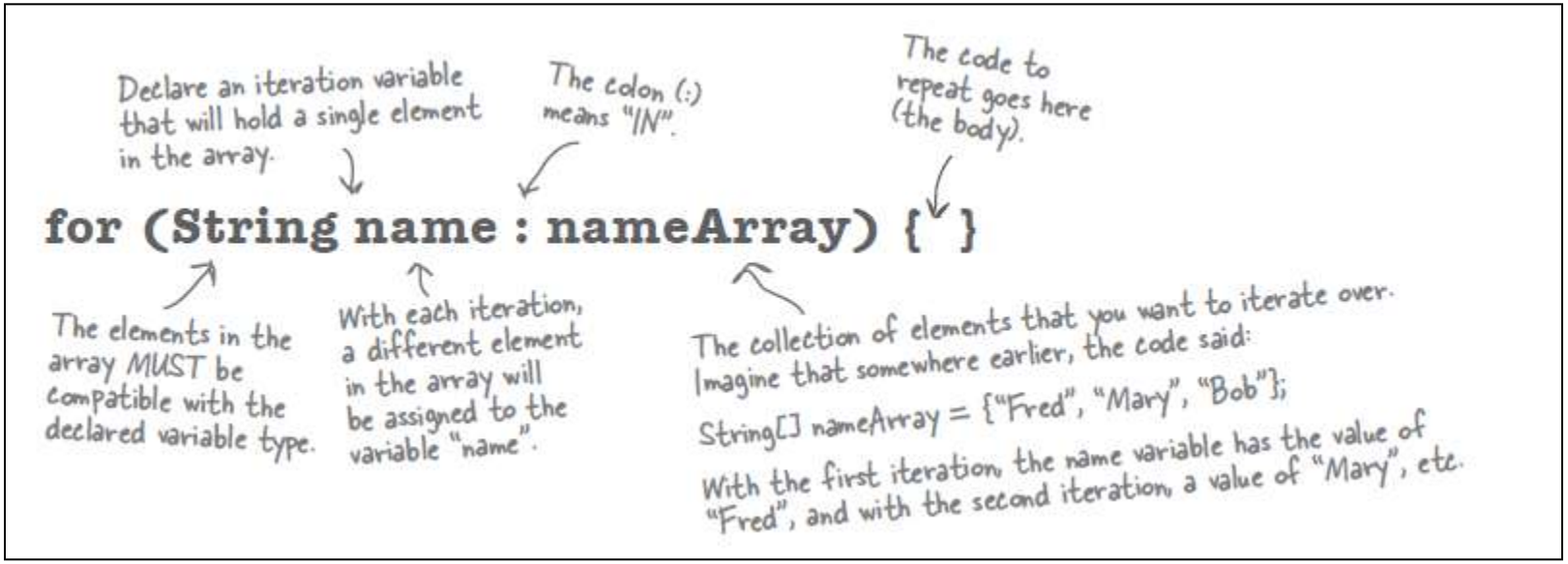
```
public class BallGoingUp
{
   public static void
   main(String [] args)
   {
      Ball b = new Ball(0.5, 0.5, 0.1);
      while (true)
      {
        b.move(0.0, 0.01);
        b.draw()
        StdDraw.show(100);
      }
   }
}
```

```
% java BallGoingUp
```



A new way to loop

- Enhanced for-loop
  - Simple way to loop over collection of things
  - Also known as a "for-each" loop
  - Works with arrays
  - Also works with collections (stay tuned)



Yippy Skippy program

- Goal: print out "skippy!" once for every command line parameters that is "yippy"
  - Use the args array and an enhanced loop
  - Be careful how you compare strings!

```
public static void main(String [] args)
{
   for (???1????)
   {
      if (???2???)
         System.out.println("skippy!");
   }
}
```

Yippy Skippy solution

- Goal: print out "skippy!" once for every command line argument that is "yippy"
  - Use the args array and an enhanced loop
  - Be careful how you compare strings!

```
public static void main(String [] args)
{
   for (String s : args)
   {
      if (s.equals("yippy"))
         System.out.println("skippy!");
   }
}
```

```
% java YippySkippy blah yippy foo yippy Yippy
skippy!
skippy!
```

Yippy Skippy, regular for-loop

- Goal: print out "skippy!" once for every command line argument that is "yippy"
  - Using a regular for-loop to do the same thing:

```
public static void main(String [] args)
{
   for (int i = 0; i < args.length; i++)
   {
      if (args[i].equals("yippy"))
         System.out.println("skippy!");
   }
}
```

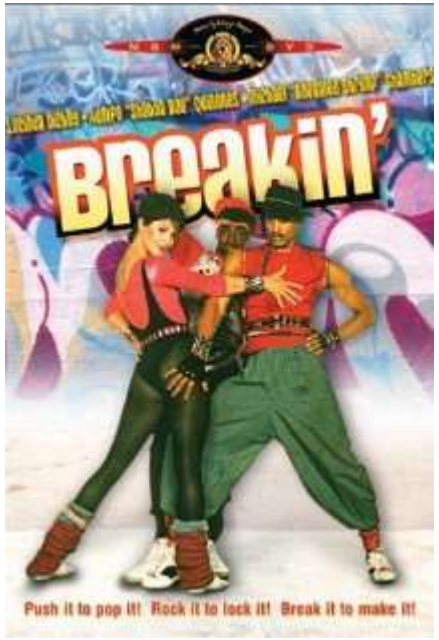
```
% java YippySkippy blah yippy foo yippy Yippy
skippy!
skippy!
```

Breaking out

- Loops normally go until loop condition false

```
int i = 0
while (i < 100)
{
   // Do some stuff
   i++;
}
```

```
for (int i = 0; i < 100; i++)
{
   // Do some stuff
}
```



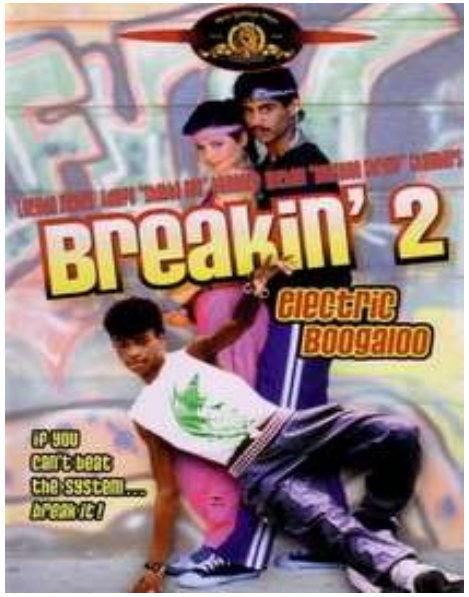
- break statement
  - Exit a loop immediately
  - No iteration, no increment, no condition
  - Straight to the code after loop

Breaking out 2

- break statement
  - Terminates enclosing loop: for, while, or do-while
  - Goal: sum `data []` array, check for invalid negative values

```
int [] data = {1, 2, 10, 20, -1, 30, 34};
int i = 0;
int sum = 0;
while (???1???)
{
  if (???2???)
     break;
  sum += data[i];
  i++;
}
if (???3???)
   System.out.println("Invalid data!");
else
   System.out.println("Sum: " + sum);
```

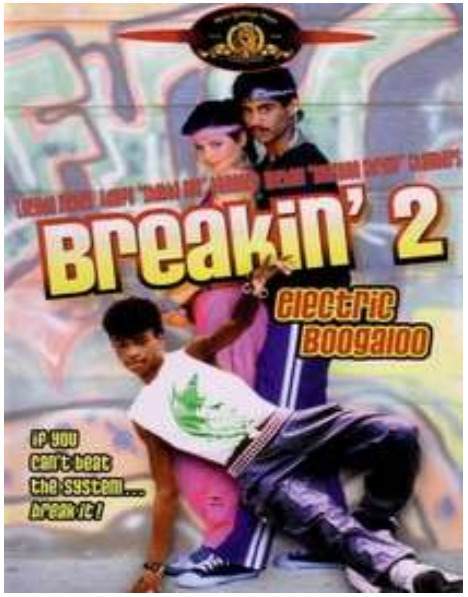
This lists tells Java how big the array should be and the initial values in each element.



Breaking out 2

- break statement
  - Terminates enclosing for-loop, while-loop, do-while loop
  - Goal: sum `data []` array, check for invalid negative values

```
int [] data = {1, 2, 10, 20, -1, 30, 34};
int i = 0;
int sum = 0;
while (i < data.length)
{
  if (data[i] < 0)
     break;
  sum += data[i];
  i++;
}
if (i < data.length)
   System.out.println("Invalid data!");
else
   System.out.println("Sum: " + sum);
```



Midterm

- Wednesday October 12th 3-5PM
- Main 205 lab
- Note sheet:
  - One-sided
  - 8 ½ x 11
  - hand-written
- No other aids, electronic or otherwise
- Covered material:
  - Lecture
  - Head First Java, chapter 1 - 5
  - Lab

Summary

- Default values
  - Local variables:
    - You need to declare initial value
  - Instance variables and arrays:
    - Java sets the default for you
  - Reference variables:
    - Default is `null`, be careful out there!
- Object oriented software
  - A collection of multiple *.java files
  - Each can have a main(), useful for testing out new class
- Enhanced for-loop
  - Good for looping over arrays of objects