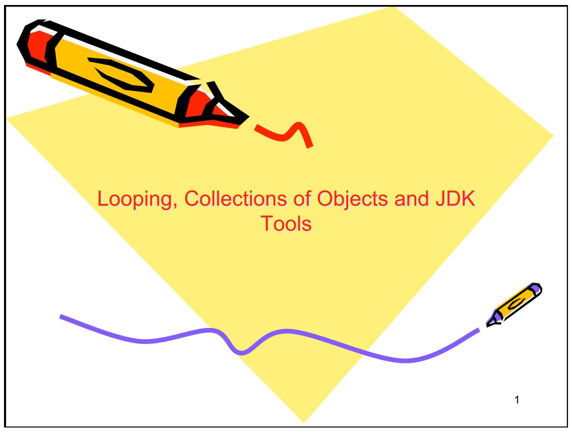
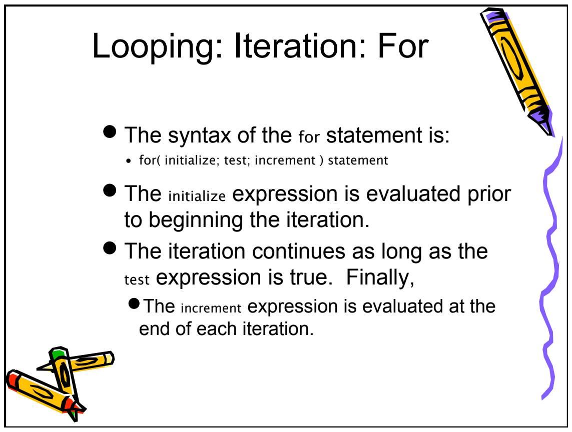


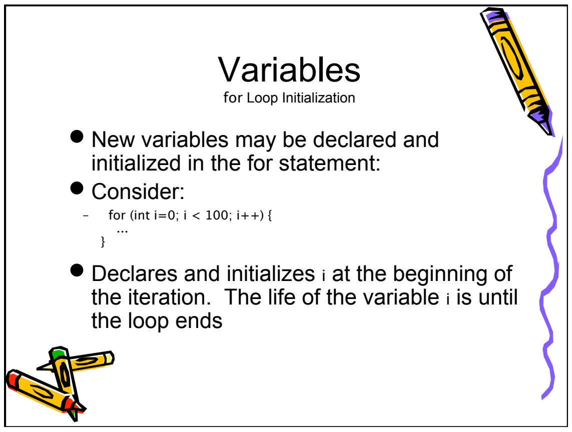
Looping, Collections of Objects and JDK Tools

1



Looping: Iteration: For

- The syntax of the for statement is:
  - for(initialize; test; increment) statement
- The initialize expression is evaluated prior to beginning the iteration.
- The iteration continues as long as the test expression is true. Finally,
  - The increment expression is evaluated at the end of each iteration.



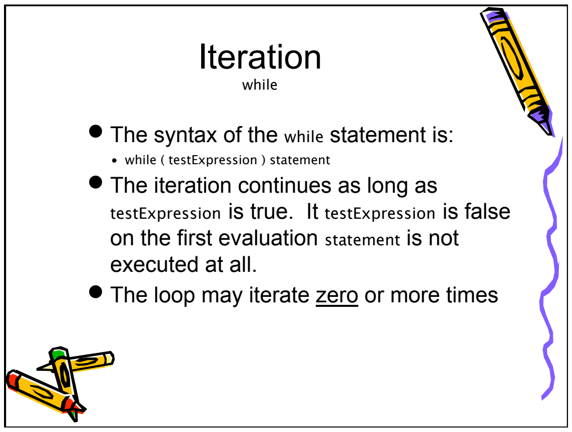
Variables

for Loop Initialization

- New variables may be declared and initialized in the for statement:
- Consider:

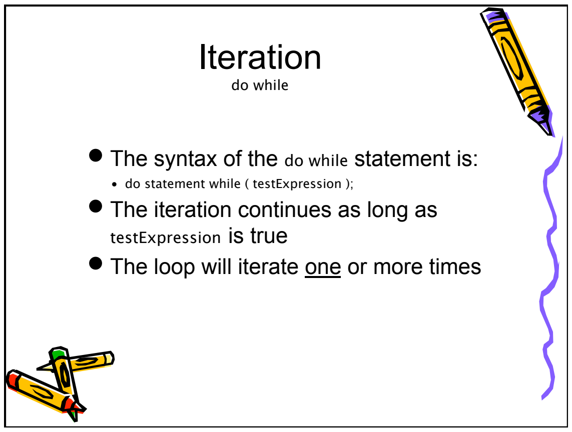
```
–    for (int i=0; i < 100; i++) {
       ...
  }
```

- Declares and initializes i at the beginning of the iteration. The life of the variable i is until the loop ends



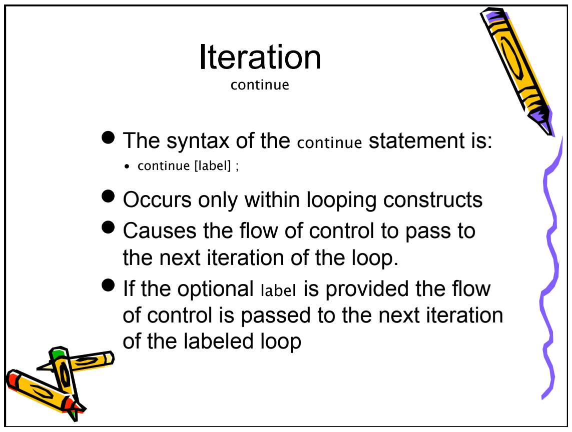
Iteration
while

- The syntax of the while statement is:
  - while (testExpression) statement
- The iteration continues as long as testExpression is true. It testExpression is false on the first evaluation statement is not executed at all.
- The loop may iterate zero or more times



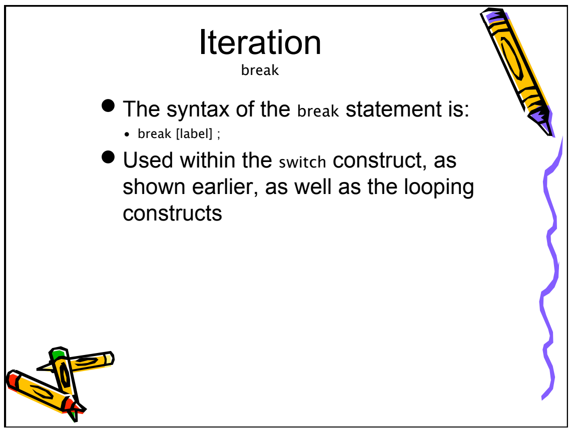
Iteration
do while

- The syntax of the do while statement is:
  - do statement while (testExpression);
- The iteration continues as long as testExpression is true
- The loop will iterate one or more times



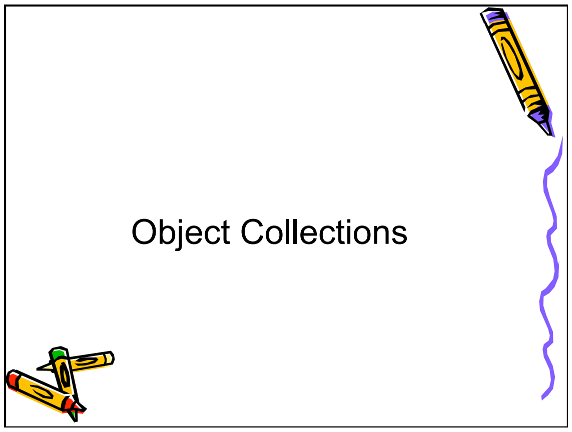
Iteration
continue

- The syntax of the continue statement is:
  - continue [label] ;
- Occurs only within looping constructs
- Causes the flow of control to pass to the next iteration of the loop.
- If the optional label is provided the flow of control is passed to the next iteration of the labeled loop

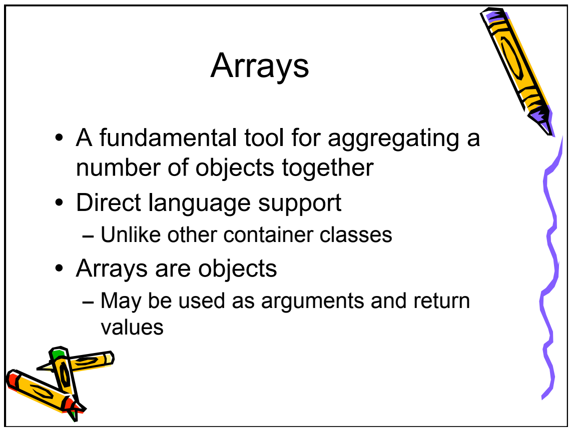


Iteration
break

- The syntax of the break statement is:
  - break [label] ;
- Used within the switch construct, as shown earlier, as well as the looping constructs

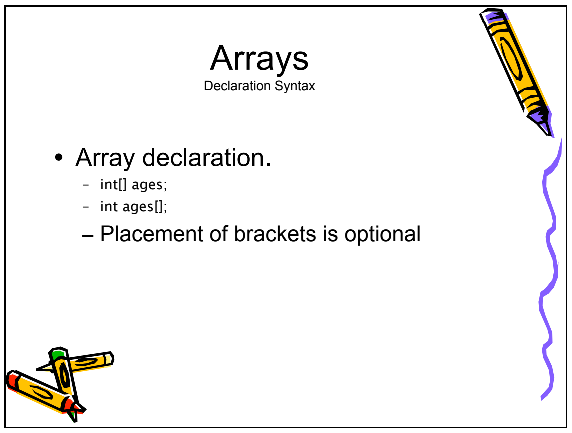


Object Collections



Arrays

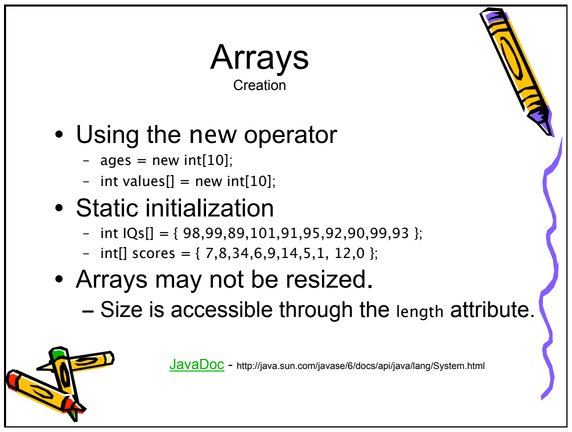
- A fundamental tool for aggregating a number of objects together
- Direct language support
  - Unlike other container classes
- Arrays are objects
  - May be used as arguments and return values



Arrays

Declaration Syntax

- Array declaration.
  - int[] ages;
  - int ages[];
  - Placement of brackets is optional

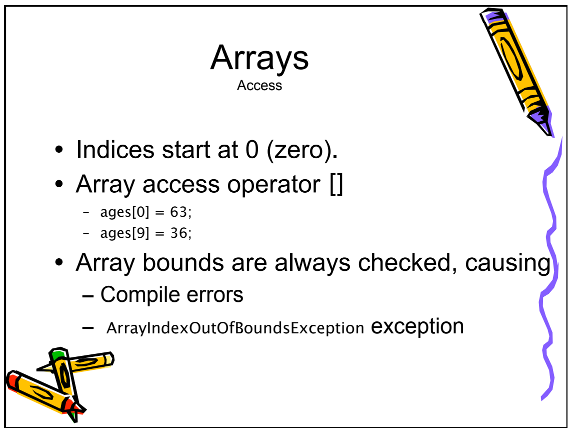


Arrays

Creation

- Using the new operator
  - ages = new int[10];
  - int values[] = new int[10];
- Static initialization
  - int IQs[] = { 98,99,89,101,91,95,92,90,99,93 };
  - int[] scores = { 7,8,34,6,9,14,5,1, 12,0 };
- Arrays may not be resized.
  - Size is accessible through the length attribute.

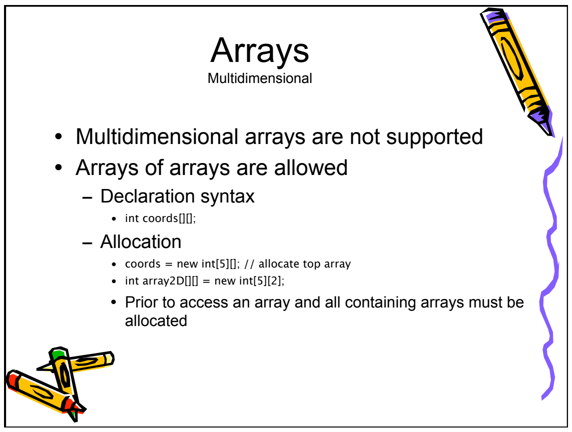
JavaDoc - http://java.sun.com/javase/6/docs/api/java/lang/System.html



Arrays

Access

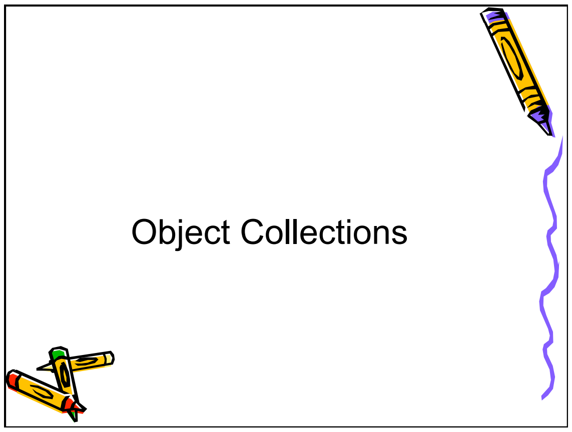
- Indices start at 0 (zero).
- Array access operator []
  - ages[0] = 63;
  - ages[9] = 36;
- Array bounds are always checked, causing
  - Compile errors
  - ArrayIndexOutOfBoundsException exception



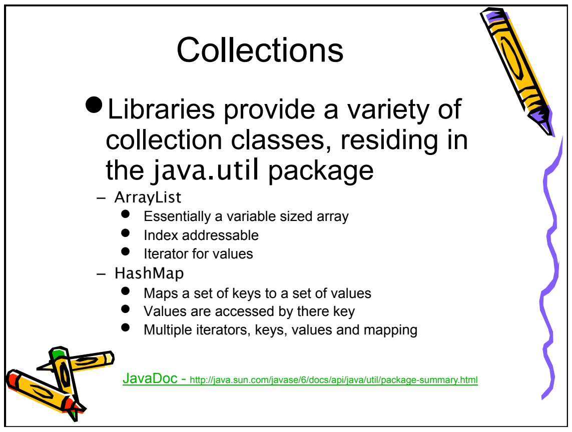
Arrays

Multidimensional

- Multidimensional arrays are not supported
- Arrays of arrays are allowed
  - Declaration syntax
    - `int coords[][];`
  - Allocation
    - `coords = new int[5][]; // allocate top array`
    - `int array2D[][] = new int[5][2];`
    - Prior to access an array and all containing arrays must be allocated



Object Collections



Collections

- Libraries provide a variety of collection classes, residing in the `java.util` package
  - ArrayList
    - Essentially a variable sized array
    - Index addressable
    - Iterator for values
  - HashMap
    - Maps a set of keys to a set of values
    - Values are accessed by there key
    - Multiple iterators, keys, values and mapping

JavaDoc - http://java.sun.com/javase/6/docs/api/java/util/package-summary.html

Using Library Classes

- Library classes must be imported using an `import` statement (except classes from `java.lang`)
- They can then be used like classes from the current project
- Packages

```
...
import java.util.ArrayList;
import java.util.Iterator;

// import java.util.*;  // frowned upon
...
```

Packages

- Namespace ID

17

ArrayList

- Methods for adding, accessing, deleting and counting its contents
  - boolean add(Object o)
  - boolean add(int index, Object o)
  - Object get(int index)
  - Object remove(int index)
  - boolean remove(Object o)
  - int size()
  - boolean isEmpty()
  - void clear()
  - Iterator iterator()

JavaDoc - http://java.sun.com/javase/6/docs/api/java/util/ArrayList.html

Iterator

- An interface for iterating over collections
- Simple set of methods:
  - boolean hasNext()
  - Object next()
  - Plus a few others…
- Usually obtained by the iterator() method

Enhanced for Loop

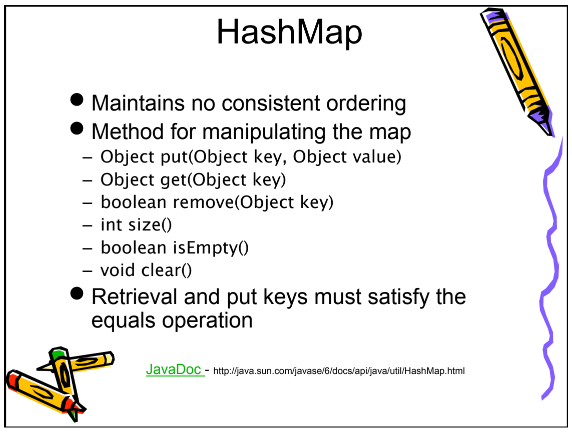
- Simplified for iteration over collections

```
ArrayList lst = new ArrayList();

for (int i = 0; i < 10; i++) {
        lst.add("Silly string. " + i * 3);
}

for (Object x : lst) {
   String s = (String)x;
      System.out.println(s);
}
```

Enhanced for Loop

- … and arrays

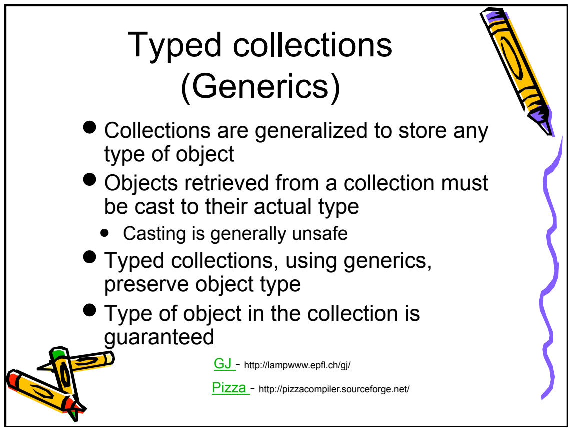
```
int[] nums = new int[9];

for (int i = 0; i < nums.length; i++) {
   nums[i] = i * 3;
}

for (int x : nums) {
     System.out.println(x);
}
```



HashMap

- Maintains no consistent ordering
- Method for manipulating the map
  - Object put(Object key, Object value)
  - Object get(Object key)
  - boolean remove(Object key)
  - int size()
  - boolean isEmpty()
  - void clear()
- Retrieval and put keys must satisfy the equals operation

JavaDoc - http://java.sun.com/javase/6/docs/api/java/util/HashMap.html

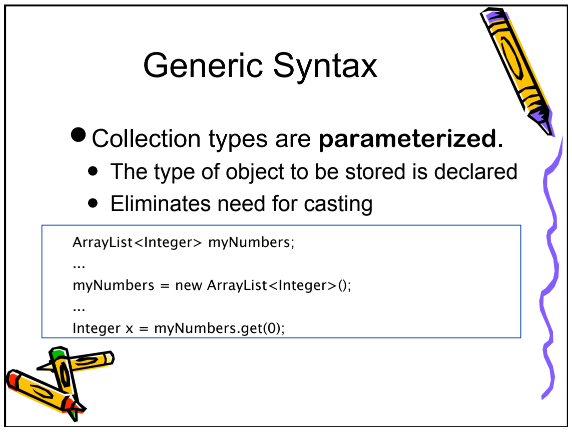


Typed collections (Generics)

- Collections are generalized to store any type of object
- Objects retrieved from a collection must be cast to their actual type
  - Casting is generally unsafe
- Typed collections, using generics, preserve object type
- Type of object in the collection is guaranteed

GJ - http://lampwww.epfl.ch/gj/

Pizza - http://pizzacompiler.sourceforge.net/



Generic Syntax

- Collection types are **parameterized.**
  - The type of object to be stored is declared
  - Eliminates need for casting

```
ArrayList<Integer> myNumbers;
...
myNumbers = new ArrayList<Integer>();
...
Integer x = myNumbers.get(0);
```

Typed Iteration

- Iterators associated with a typed collection share the type

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

for (int i = 0; i < 10; i++) {
        nums.add(new Integer(i * 3));
}

for (Integer x : nums) {
     System.out.println(x);
}
```

Typed HashMaps

- HashMaps operate with (key, value) pairs.
- A typed HashMap requires two type parameters

See Example HashMashTypedExample

Primitive Classes

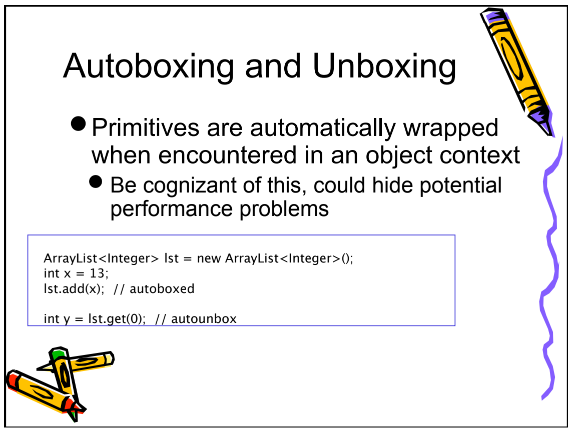
•ArrayLists hold objects – what about the primitive types?

–Each of the primitive types has a corresponding class

–The "primitive" objects are immutable

Primitive	Class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

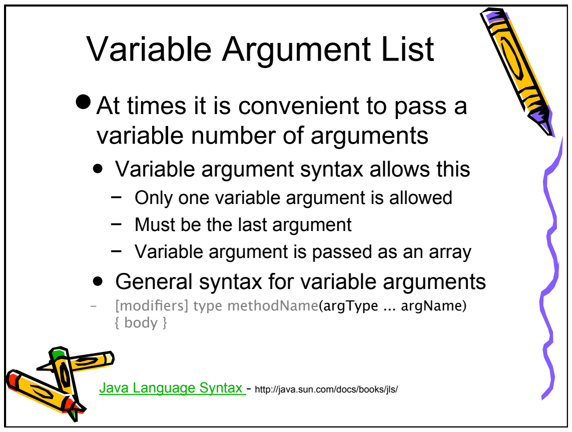
JavaDoc - http://java.sun.com/javase/6/docs/api/java/lang/package-summary.html



Autoboxing and Unboxing

- Primitives are automatically wrapped when encountered in an object context
  - Be cognizant of this, could hide potential performance problems

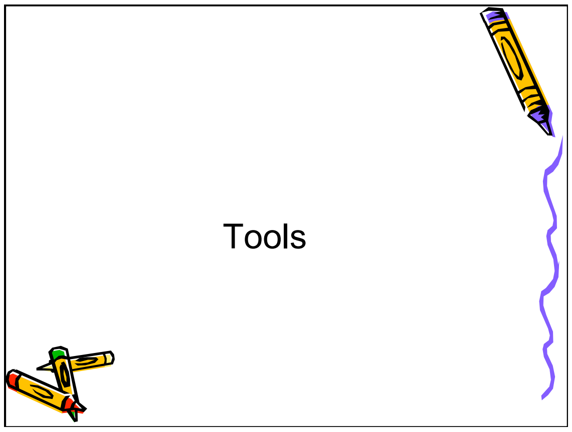
```
ArrayList<Integer> lst = new ArrayList<Integer>();
int x = 13;
lst.add(x);  // autoboxed

int y = lst.get(0);  // autounbox
```



Variable Argument List

- At times it is convenient to pass a variable number of arguments
  - Variable argument syntax allows this
    - Only one variable argument is allowed
    - Must be the last argument
    - Variable argument is passed as an array
  - General syntax for variable arguments
    - [modifiers] type methodName(argType ... argName) { body }

Java Language Syntax - http://java.sun.com/docs/books/jls/



Tools

Java Development Kit (JDK)

- Free from Sun Microsystems
- Includes
  - javac (Java compiler)
  - java (Java interpreter/virtual machine)
  - javadoc (Document extractor/generator)
  - Plus other tools…

Java Tools - http://java.sun.com/javase/6/docs/technotes/tools/index.html

JDK
javac

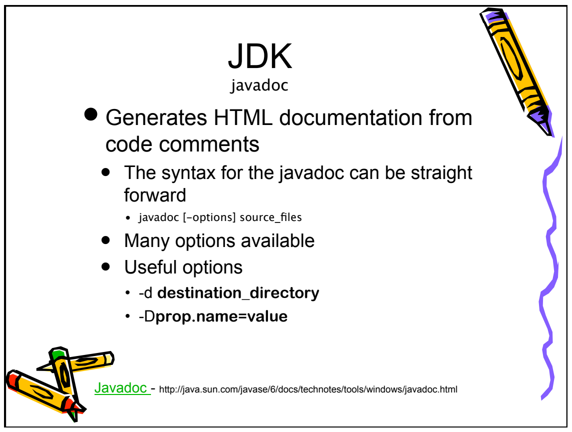
- Generates a ".class" file for each class within the file being compiled
- Java source files are expected to have a ".java" extension
  - The syntax for the Java compiler is:
    - javac [-options] files
  - Popular options
    - -classpath **classpath_specification**
    - -d **destination_directory**

Javac - http://java.sun.com/javase/6/docs/technotes/tools/windows/javac.html

JDK
java

- Executes programs generated by the compile process
  - The syntax for the JVM is straight forward
    - java [-options] classname [arguments]
  - Popular options
    - -classpath **classpath_specification**
    - -D**prop.name=value**
  - The arguments are provided as arguments to the program being executed

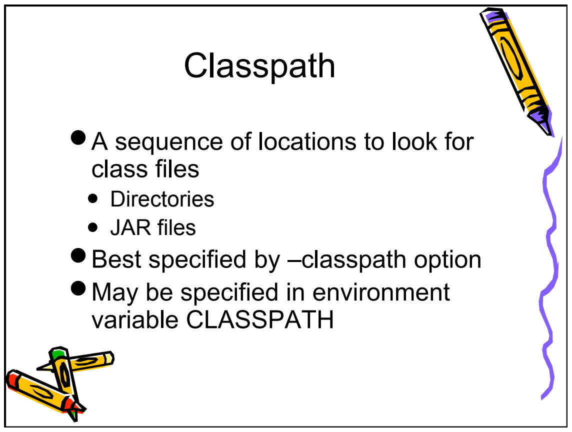
Java - http://java.sun.com/javase/6/docs/technotes/tools/windows/java.html



JDK
javadoc

- Generates HTML documentation from code comments
  - The syntax for the javadoc can be straight forward
    - javadoc [–options] source_files
  - Many options available
  - Useful options
    - -d **destination_directory**
    - -D**prop.name=value**

Javadoc - http://java.sun.com/javase/6/docs/technotes/tools/windows/javadoc.html



Classpath

- A sequence of locations to look for class files
  - Directories
  - JAR files
- Best specified by –classpath option
- May be specified in environment variable CLASSPATH