CSPP 511-01: 
Introduction to Object-Oriented Programming 

Harri Hakula 

Ryerson 256, tel. 773-702-8584 

hhakula@cs.uchicago.edu 

July 5, 2000
Outline

• Branching and Loops
• Examples
• Arrays
Branching and Loops

Branching:
- if-else
- switch

Loops:
- while
- do-while
- for
for Statement Again

for (Initializing Action; Boolean Expression; Action After Each Iteration)

Body

is equivalent to (Note the added block!)

{

  Initializing Action;

  while (Boolean Expression)

  {
    Body
    Action After Each Iteration;
  }

}
Even More on for Statement

The initialization and iteration parts of a for loop can be comma-separated lists of expressions.

```java
for(int i=0, j=0; i < MAX; ++i, ++j) { }
```

is legal, but

```java
for(int i = 0, double j = 0; i < MAX;
     ++i, j += 1.0) { }
```

is not, since comma-separated initializations must be of the same type.
Yet Another Slide on for Statement

Any of the three parts can be omitted. If the boolean expression is omitted, it is considered to be true. Therefore we have two popular idioms for infinite loops:

```java
for(;;) { ... }
while(true) { ... }
```
Examples: Fibonacci

```java
public class Fibonacci {
    public static void main(String args[]) {
        int lo = 1;
        int hi = 1;
        System.out.println(lo);
        while(hi < 50) {
            System.out.println(hi);
            hi = lo + hi;
            lo = hi - lo;
        }
    }
}
```
public class Fibonacci {
    static final int MAX = 70;
    public static void main(String args[]) {
        int lo = 1;
        int hi = 1;
        System.out.println(lo);
        while(hi < MAX) {
            System.out.println(hi);
            hi = lo + hi;
            lo = hi - lo;
        }
    }
}
Examples: Fibonacci 3

```java
public class Fibonacci {
    static final int MAX = 50;
    public static void main(String args[]) {
        int lo = 1, hi = 1;
        String mark;
        System.out.println(lo);
        while(hi < MAX) {
            if(hi % 2 == 0) mark = " *";
            else mark = ""
            System.out.println(hi + mark);
            hi = lo + hi;
            lo = hi - lo;
        }
    }
}
```
Example: Fibonacci 4

```java
public class Fibonacci {
    static final int MAX_INDEX = 9;
    public static void main(String args[]) {
        int lo = 1, hi = 1;
        String mark;
        System.out.println("1: " + lo);
        for(int i = 2; i <= MAX_INDEX; ++i) {
            if(hi % 2 == 0) mark = " *";
            else mark = ""
            System.out.println(i+: " +hi+mark);
            hi = lo + hi;
            lo = hi - lo;
        }
    }
}
```
Arrays

Every JAVA program has at least one array – String[] args. Arrays are simple collections of objects of the same type. Every element of args is an instance of String.

Declaration: String[] args; String args[]; are equivalent.

Every element of an array has its own index. The length of an array is stored in a data member length. For example:

int sz = args.length;
Arrays Cntd.

```java
public class CommandLine {
    /* Print out the command line arguments.
     * java CommandLine one two three four
     */
    public static void main(String[] args) {
        for(int i = 0; i < args.length; ++i) {
            System.out.print(args[i] + " ");
        }
        System.out.println();
    }
}
```
Creating Arrays

In JAVA the name of a variable is just a reference to a memory location. So the declaration `int[] ary;` means that `ary` can point to a collection of ints, but the size of the array is yet to be determined.

Arrays are initialized using the `new` operator:

```java
int[] ary = new int[3];
int[] arh;
arh = new int[1001];
```
Creating Arrays Cntd.

Initialization can be explicit (size is determined automatically):

```java
String[] teams = { "Cubs", "White Sox" };
String[] teams = new String[]{ "Cubs", "White Sox" };
```

or done element by element

```java
String[] teams = new String[3];
teams[0] = "Cubs";
teams[1] = "White Sox";
```