• Arrays
• The Class Java.lang.Boolean
• boolean and Integer Operators

Overview
- The integer bitwise operators & (and) and ~
- The bitwise complement operator ~
- The signed and unsigned shift operators <<, >>, and <<<
- The decrement operator --, both prefix and postfix
- The increment operator ++, both prefix (++x) and postfix (x++)
- The additive operators + and -
- The multiplicative operators *, /, and %
- The unary plus and minus operators and

The numerical operators, which result in a value of type int:

- = The numerical equality operators == and !=
- The numerical comparison operators <, <=, >, >=, and

The comparison operators, which result in a value of type boolean:

Integer Operators
Boolean Operators

- The conditional-and and conditional-or operators && and ||
- The logical operators & and | and !
- The logical-complement operator ~
- The relational operators == and =
Programming Tips

• Don't limit your use of Boolean expressions to if statements.

boolean flag = x > y;

return true;
else
return false;

if (x > y)

Don't —

boolean flag = false;
else
flag = true;

if (x > y)

Don't —

return x >= y;

— Do

— Do

February 28, 2000
... 

(((\y > x) \implies \text{flag}) \implies \text{flag}) \implies \text{flag}) \implies \text{flag}) \implies \text{flag}) 

... 

... 

\text{Do} 

... 

Choose carefully between \& and \&\&.

Programming Tips (cont.)
... $(\exists y \, x > y)$  $(\forall y \, x > y)$

- Do
- Don't

Avoid pointless terms.

Programming Tips (cont.)
```java
public static boolean getBoolean(String name, boolean value) {
    return value;
}

public static boolean getBooleanValue(String s) {
    return false;
}

public boolean booleanValue() {
    return false;
}

public int hashCode() {
    return 0;
}

public boolean equals(Object obj) {
    return true;
}

public String toString() {
    return "true";
}

public String toString(boolean s) {
    return "false";
}

public boolean booleanValue(boolean value) {
    return true;
}

public static final boolean FALSE = new Boolean(false);

public static final boolean TRUE = new Boolean(true);
```

- Objects of type `Boolean` represent primitive values of type `boolean`.

**The Class** `java.lang.Boolean`
Arrays

Arrays

Arrays

Arrays

Arrays

Arrays

Arrays

Arrays

Arrays

Arrays

Arrays

Arrays

Arrays

Arrays
reference, including interface types and abstract class types.

- The element type of an array may be any type, whether primitive or
  An array’s length is not part of its type.

- An array corresponding to a one-dimensional array of integers
  or more empty pairs of square brackets. For example, int[] is the type
  An array type is written as the name of an element type followed by one

Array Types
Array Variables

Type may contain references to arrays of different lengths.

Because an array's length is not part of its type, a single variable of array

creates the variable, which can contain a reference to an array.

Declaring a variable of array type does not create an array object. It only

precision floating-point numbers.

declares a variable whose type is a two-dimensional array of double-

type. For example, the array's type followed by the array's name. For example,

Array variables are declared like other variables: a declaration consists of

double[] matrixDoubles;

Array Variables
• Please use the latter:

```c
double[] matrix;
double[] rowvector, colvector;
```

or

```c
double[] rowvector, colvector, matrix[];
```

• This declaration is equivalent to

```c
double rowvector[], colvector[], matrix[];
```

• Example:

To complete declarations, C/C++-like syntax is also supported, for

Array Variables (cont.)
Another way to initialize array variables is:

```java
String[] arrayStrings = new String[10];
```

To hold a reference to an array object with room for ten references, declares a variable whose type is an array of strings, and initializes it:

```java
String[] arrayOfStrings = new String[10];
```

Array objects, like other objects, are created with new. For example:

Array Creation
would print "5."

```java
System.out.println(array[0].length);

int[] array[0] = { 1, 2, 3, 4, 5 };
```

Example:

- The array's length is available as a final instance variable `length`.

```java
int[] array[0] = {{1,2},{3,4}};
```

- Once an array object is created, its never changes length.

Array Creation (cont.)
Array Accesses

causes an IndexOutOfBoundsException to be throw.
that is less than zero or greater than or equal to the length of the array
All array accesses are checked at run time. An attempt to use an Index
catch (ArrayStoreException e) {
    pa[0] = new Point();
}
try
System.out.println(pa[1] == null);
new ColorPoint[10];
    cpa = ColorPoint[]
}

\begin{itemize}
\item Consider
\end{itemize}

\textbf{Array Store Exception}
Java. J. 

true

produces the output:

Array Store Exception (cont.)