Web Technology 5
Java Basics

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

Primitive Data types

Primitive Data types

- byte: is a signed 8-bit integer
  - byte b;
- short: is a signed 16-bit integer
  - short s;
- int: is a signed 32-bit integer
  - int i;
- long: is a signed 64-bit integer
  - long l;
- float: is a single-precision 32-bit floating point number
  - float f;
- double: is a double-precision 64-bit floating point number
  - double d;
- char: is a single 16-bit Unicode character
  - char c;
- boolean: is a data type having only 2 values: true and false
  - boolean b;

Primitive Data types...

Primitive Data types...

- Java is a strongly-typed language
- Every variable and expression has a type
- Every type is strictly defined
- All assignments are checked for type-compatibility
- No automatic conversion of non-compatible, conflicting types
- Java compiler type-checks all expressions and parameters
- Any typing errors must be corrected for compilation to succeed

Elements of Java Program

Elements of Java Program

- Whitespaces: is a space, or tab or a newline character
  - class HelloWorld
- Identifiers: used to identify variables, methods and classes. They can be a sequence of alphabets, numbers, underscore character and a dollar-sign character. It can't begin with a digit
  - Identifiers are case- sensitive
  - int e, E;
- Literals: used to specify constant values in a java program
  - 100
  - 300.56
  - 'M'
  - Ïndia"

Elements of Java Program...

Elements of Java Program...

- Comments:
  - Single-line Comments:
    //end of the loop

    /*end of the loop*/
  - Multiline Comments:
    /*this is a multiline comments
    of two lines*/
  - Documentation Comments:
    /** documentation */
- Separators:
  - Semicolon (;)
  - Period (.)
  - Comma (,)
  - Brackets ([ ])
  - Curley braces ({ })
  - Parentheses (( ))

Elements of Java Program...

Elements of Java Program...

Keywords: reserved words in java:

abstract	assert	boolean	break	byte	case
catch	char	class	const	continue	default
do	double	else	extends	final	finally
float	for	if	implements	import	instanceof
int	interface	long	native	new	package
private	protected	public	return	short	static
strictfp	super	switch	synchronize	this	throw
throws	transient	try	void	volatile	while

Elements of Java Program...

Elements of Java Program...

- Escape sequences:
  - \ddd: octal character ddd
  - \uxxxx: hexadecimal Unicode character xxxx
  - \**'**: single quote
  - \**"**: double quote
  - \\ \\: backslash
  - \r: carriage return
  - \n: new line
  - \f: form feed
  - \t: tab
  - \b: backspace

Variables in Java

Variables in Java

- All variables must be declared before they can be used
  - ***datatype identifier [=value];***
  - char c='A';
- It is also possible to initialize a variable dynamically at runtime
  - double area= length * breadth;
- Scope and Lifetime of Variables:
  - Scope determines the visibility of program elements with respect to other program elements
  - Block defines a scope. A block starts with a '{' and ends with a '}'
  - Each time a new block is started, a new scope begins
  - The life of the variable is within its scope

Automatic conversion of compatible data types

Automatic conversion of compatible data types

- If the data type of the value and the data type of the variable are compatible and the data type of the variable is larger than that of the value, then java will automatically allow the conversion
- byte $\rightarrow$ short, int, long, float, double
- short $\rightarrow$ int, long, float, double
- int $\rightarrow$ long, float, double
- long $\rightarrow$ float, double
- float $\rightarrow$ double
- char $\rightarrow$ int, long, float, double

Type Casting

Type Casting

- If the data type of the value is larger than the data type of the variable to which it is being assigned, then cast is used to explicitly convert the data type of the value to the data type of the variable
  - **(targetType) value**
  - double d=12.34D;
  - float f=(float) d;
- *If the whole number is too large to fit into the target integer type, the value will be reduced modulo the target type's range*

Promotion of data types in expressions

- byte and short are always promoted to int
- if one operand is long or float or double, the whole expression is promoted to long or float or double respectively

Operators

Operators

- Operators are used to build value expressions
  - Unary
  - Binary
  - Ternary
- **Arithmetic** Operator
  - +, -, *, /, %
- **Relational** Operator
  - Outcome is always a value of type Boolean
  - ==, $\neq$, < , $\leq$, > , $\geq$
- **Logical** Operator
  - Logical operators act upon Boolean operands only
  - &, |, !, $\wedge$
- **Short Circuit Logical** Operator
  - If these operators are used, java will not evaluate the second operand if the result can be determined by the first operand alone
  - & &, ||

Operators...

Operators...

- **Increment** & **Decrement** Operator
  - The operand must be a numerical variable
  - prefix version evaluates the value of the operand after performing the increment/decrement operation
  - postfix version evaluates the value of the operand before performing the increment/decrement operation
- **Bitwise Logical** Operator
  - $\&, |, \sim, \wedge, < <, > >, > > >$
- **Assignment** Operator
  - Types of the variable and expression must be compatible
  - =
- Other Operators
  - Conditional operator (? :)
  - $\sqsubset \; \sqsupset$
  - .
  - (params)
  - (type)
  - new
  - instanceof

Operator Precedence

Operator Precedence

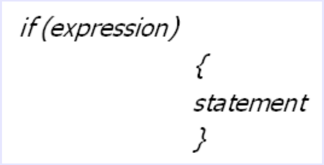
When operators have the same precedence, the earlier one binds stronger

highest			
()	[]	.	
++	--	~	!
*	/	%	
+	-		
>>	>>>	<<	
>	>=	<	<=
==	!=		
&			
^			
|			
&&			
||			
?:			
=	op=		
lowest			

Selection Statements

Selection Statements

- Java selection statements allow to control the flow of program's execution based upon conditions known only during run-time
- if statement

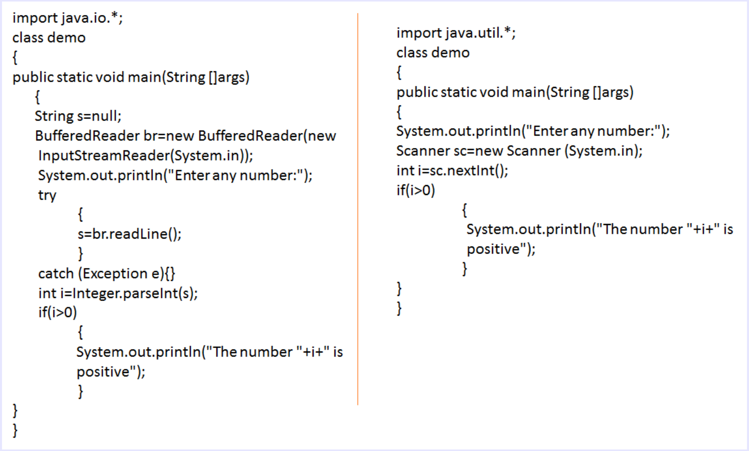


```
if (expression)
                {
                statement
                }
```

  - The expression must be of type Boolean

Selection Statements...

Q: Input a number and check whether it is positive



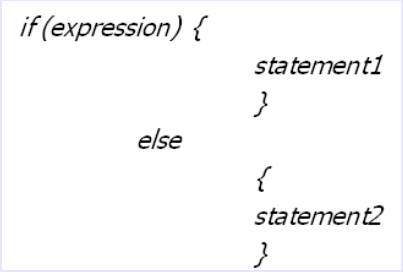
```
import java.io.*;
class demo
{
public static void main(String []args)
    {
    String s=null;
    BufferedReader br=new BufferedReader(new
     InputStreamReader(System.in));
     System.out.println("Enter any number:");
     try
            {
            s=br.readLine();
            }
     catch (Exception e){}
     int i=Integer.parseInt(s);
     if(i>0)
            {
            System.out.println("The number "+i+" is
            positive");
            }
}
}
```

```
import java.util.*;
class demo
{
public static void main(String []args)
{
System.out.println("Enter any number:");
Scanner sc=new Scanner (System.in);
int i=sc.nextInt();
if(i>0)
          {
           System.out.println("The number "+i+" is
           positive");
          }
}
}
```

Selection Statements...

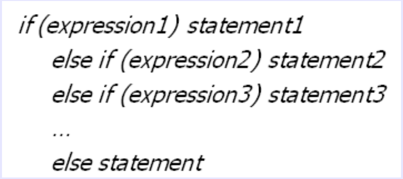
Selection Statements...

- if-else statement



```
if (expression) {
                     statement1
                     }
          else
                     {
                     statement2
                     }
```

- if-else-if statement

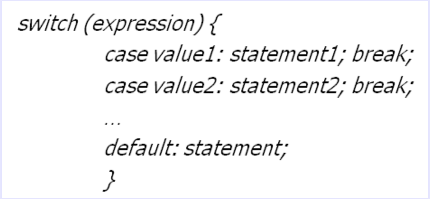


```
if (expression1) statement1
   else if (expression2) statement2
   else if (expression3) statement3
   ...
   else statement
```

Switch Statement

Switch Statement

- switch provides a better alternative than if-else-if when the execution follows several branches depending on the value of an expression



```
switch (expression) {
            case value1: statement1; break;
            case value2: statement2; break;
            ...
            default: statement;
            }
```

- *Expression must be of type byte, short, int or char*
- *Each of the case values must be a literal of the compatible type*
- *Case values must be unique*
- *Break makes sure that only the matching statement is executed*

Iteration Statements

Iteration Statements

- Java iteration statements enable repeated execution of part of a program until a certain termination condition becomes true
- while statement

```
while (expression)
                    statement
```

- do-while statement

```
do
          statement
          while (expression);
```

- for statement

```
for (initialization; termination; increment)
                  statement
```

Jump Statements

Jump Statements

- Java jump statements enable transfer of control to other parts of program
- break statement
  - *break;*
  - Java does not have goto statement
  - *break label;*
  - *label: { ... }*
- continue statement
  - The break statement terminates the block of code, in particular it terminates the execution of an iterative statement
  - The continue statement forces the early termination of the current iteration to begin immediately the next iteration
  - *continue;*
  - *continue label;*

Math Class

Math Class

- Math class contains all the floating-point functions that are used for geometry and trigonometry
  - Math.sin()
  - Math.sinh()
  - Math.cbrt(), Math.exp(), Math.log(), Math.log10(), Math.pow(), Math.sqrt()
  - Math.abs(), Math.ceil(), Math.floor()
  - Math.max(), Math.min(), Math.round()
  - Math.random()
  - Math.toDegrees(), Math.toRadians()
  - Math.PI