JAVA PROGRAMMING BASICS

By Pakita Shamoi

Identifiers

- **An identifier is a name given to a package, class, interface, method, or variable. It allows a programmer to refer to the item from other places in the program.**
- An identifier must start with a letter, an underscore, or a dollar sign.
- An identifier cannot contain operators, such as +, -, and so on.
- An identifier cannot be a reserved word (e.g. int)
- An identifier cannot be `true`, `false`, or `null`.
- An identifier can be of any length.

Declaring Variables


```
int x;          // Declare x to be an
                // integer variable;
double radius; // Declare radius to
                // be a double variable;
char a;         // Declare a to be a
                // character variable;
```

Assignment Statements

- `x = 1;  // Assign 1 to x;`
- `radius = 1.0;  // Assign 1.0 to radius;`
- `a = 'A';  // Assign 'A' to a;`

```
radius = 1.0;
area = radius*radius*3.14159;
System.out.println("The area is “ + area + " for
radius "+radius);
```

- Declaring and Initializing in 1 step
  - `int x = 1;`
  - `double d = 1.4;`

Constants


```
final datatype CONSTANTNAME = VALUE;
```

- `final double PI = 3.14159;`
- `final int SIZE = 3;`

Numerical Data Types

`byte`	8 bits
`short`	16 bits
`int`	32 bits
`long`	64 bits
`float`	32 bits
`double`	64 bits

Operators

- **+, -, *, /, and %**

- 5/2 yields an integer 2.
- 5.0/2 yields a double value 2.5

- 5 % 2 yields 1 (the remainder of the division)

Shortcut Operators

Operator	*Example*	*Equivalent*
`+=`	`i+=8`	`i = i+8`
`-=`	`f-=8.0`	`f = f-8.0`
`*=`	`i*=8`	`i = i*8`
`/=`	`i/=8`	`i = i/8`
`%=`	`i%=8`	`i = i%8`

Increment and Decrement Operators

- `x = 1;`
- `y = 1 + x++;` What is the value of y?
- `y = 1 + ++x;` What is the value of y?

- `y = 1 + x--;`
- `y = 1 + --x;`

What is the difference between x-- and --x?

Numeric Type Conversion

Consider the following statements:

```
byte i = 100;
long myLong = i*3+4;
double d = i*3.1+l/2;

int x = myLong; (Wrong)
long l = x; (fine, implicit casting)
```

Type Casting

Implicit casting

```
double d = 3;
```

Explicit casting

```
int i = (int)3.0;
```

What is wrong? int x = 5/2.0;

The boolean Type and Operators


```
boolean lightsOn = true;

boolean lightsOn = false;
```

- && (and) (1 < x) && (x < 100)
- || (or) (lightsOn) || (isDayTime)
- ! (not) !(isStopped)

Programming Style and Documentation

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines

Appropriate Comments

- Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

- Include your name, class section, instruction, date, and a brief description at the beginning of the program.

Naming Conventions

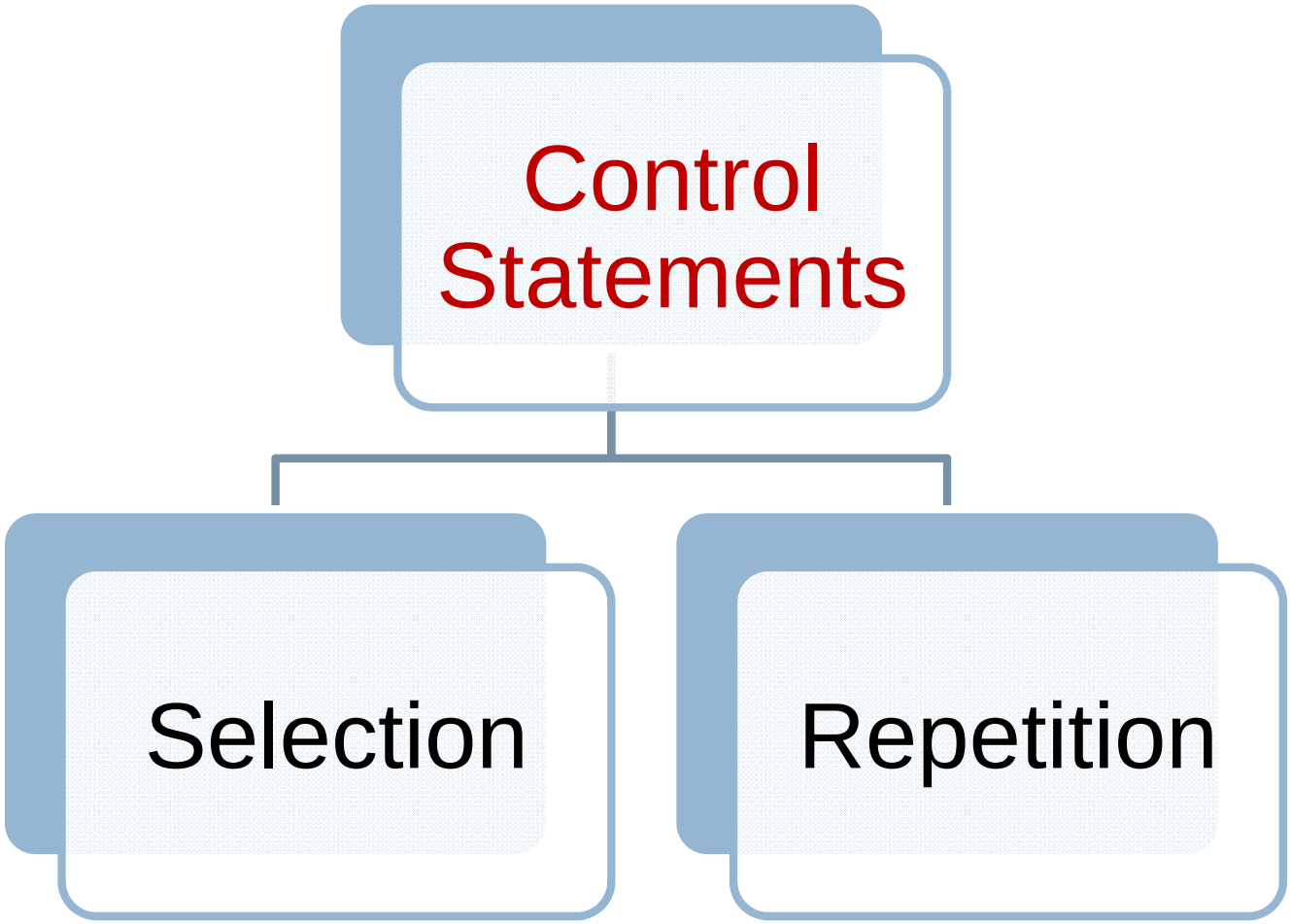
- Choose meaning and descriptive names.
- Variables and method names:
  - Use lowercase. If the name consists of several words, concatenate all in one, use lowercase for the first word, and capitalize the first letter of each subsequent word in the name. For example, the variables `radius` and `area`, and the method `computeArea`.
- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name `MyCircle`.
- Constants:
  - Capitalize all letters in constants. For example, the constant `PI`

Programming Errors

- Syntax Errors
  - Detected by the compiler.
- Runtime Errors
  - Causes the program to abort
- Logic Errors
  - Produces incorrect result

Think of examples for each error type

Control statements



Selection Statements

- `if` Statements
- `switch` Statements
- Conditional Operators

if , if…else statements

- if (booleanExpression)

```
{
  statement(s);
}
```

- if (booleanExpression)

```
{
  statement(s)-for-the-true-case;
}
else
{
  statement(s)-for-the-false-case;
}
```

`if...else` Example


```
if (radius >= 0)
{
  area = radius*radius*PI;
  System.out.println("The area for the “
    + “circle of radius " + radius +
    " is " + area);
}
else
{
  System.out.println("Negative input");
}
```

Conditional Operator


```
if (x > 0) y = 1
else y = -1;

is equivalent to

y = (x > 0) ? 1 : -1;
```

`switch` Statements


```
switch (year)
{
  case 7:  annualInterestRate = 7.25;
           break;
  case 15: annualInterestRate = 8.50;
           break;
  case 30: annualInterestRate = 9.0;
           break;
  default: System.out.println(
   "Wrong number of years, enter 7, 15, or 30");
}
```

Repetitions

- `while` Loops
- `do ...while` Loops
- `for` Loops
- `break` and `continue`

while, do…while Loops


```
while (continue-condition)
{
 // loop-body;
}
```

```
do
{
  // Loop body;
} while (continue-condition)
```

for Loops


```
for (control-variable-initializer;
      continue-condition; adjustment-statement)
{
   //loop body;
}
```

Example:

```
for (int i = 0; i<100; i++)
{
  System.out.println("Welcome to Java! ” + i);
}
```

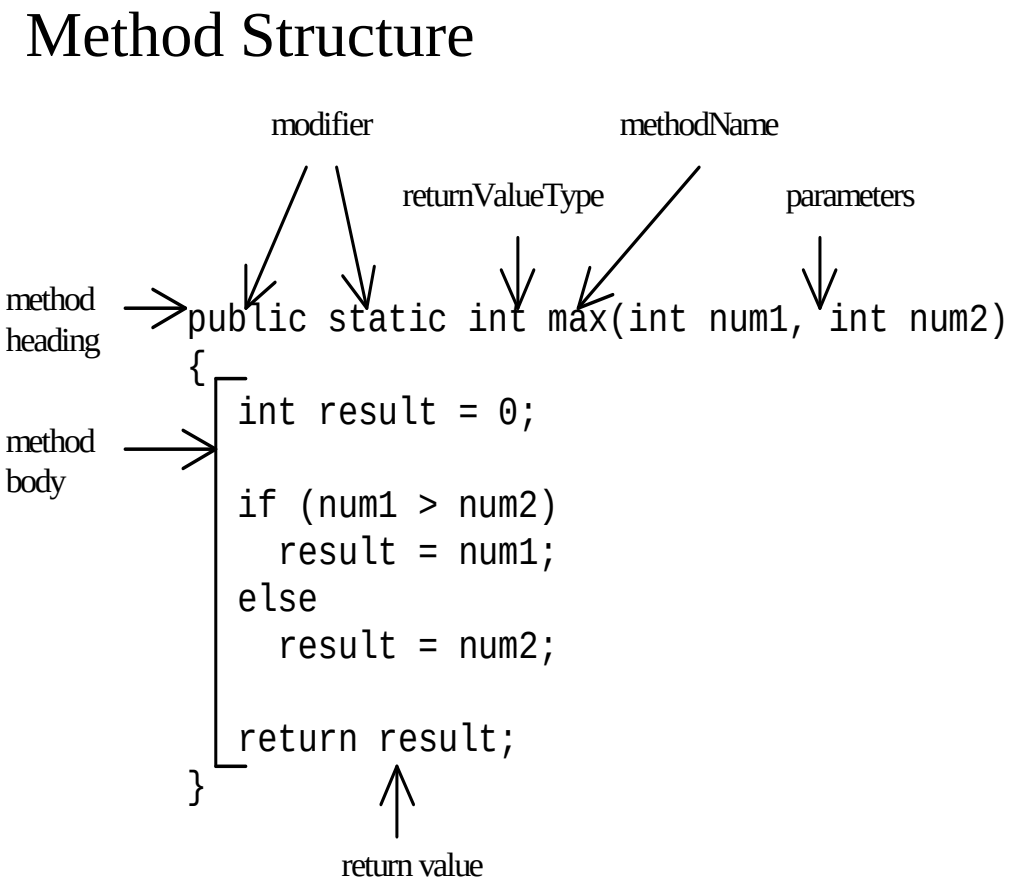
- break
- continue

Methods

- Introducing Methods
- Declaring Methods
- Calling Methods
- Pass by Value
- Overloading Methods
- Method Abstraction
- The `Math` Class

Introducing Methods

A method is a collection of statements that are grouped together to perform an operation.



Declaring Methods


```
public static int max(int num1, int num2)
{
  if (num1 > num2)
    return num1;
  else
    return num2;
}
```

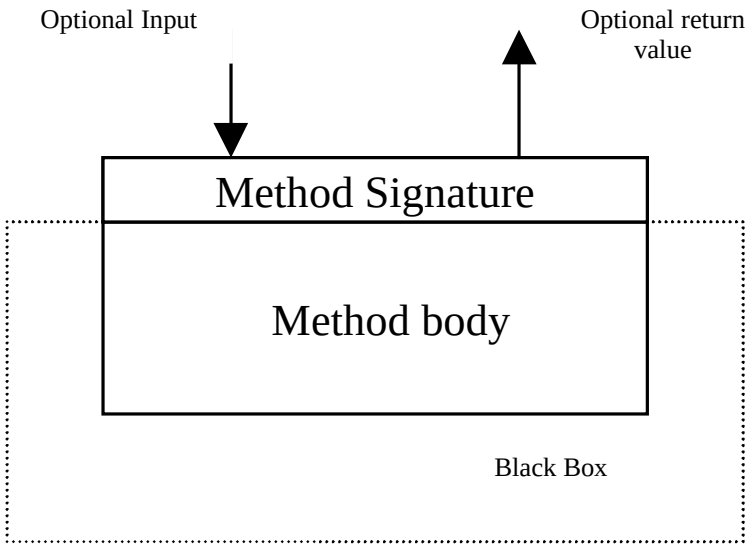
Overloading Methods

Overloading the `max` Method

```
double max(double num1, double num2)
{
  if (num1 > num2)
    return num1;
  else
    return num2;
}
```

Method Abstraction

You can think of the method body as a black box that contains the detailed implementation for the method.



The `Math` Class

- Class constants:
  - `PI, E`
- Class methods:
  - Trigonometric Methods
  - Exponent Methods
  - Miscellaneous

  - Pow, log, sqrt, sin, cos, abs, random, max, min etc.

The `String` Class

- Declaring a String:
  - `String message = "Welcome to Java!"`
  - `String message = new String("Welcome to Java!“);`
- String Comparisons
  - `if (s1.equals(s2)){ // s1 and s2 have the same contents }`
  - `if (s1 == s2) { // s1 and s2 have the same reference }’`
- String Concatenation

`String` is an immutable class, its values cannot be changed individually.

```
String s1 = "Welcome to Java";
String s2 = s1.substring(0,10) + "HTML";
String s3 = s1.contact(s2);
String s3 = s1 + s2;
```

String Concatenation

- String Length

```
message = "Welcome";
message.length() (returns 7)
```

- Retrieving Individual Characters in a String
  - Do not use `message[0]`
  - Use `message.charAt(index)`
  - Index starts from `0`