Java

An introduction

Example 1

```java
public class Example1 {
    public static void main(String[] args) {
        System.out.println("This is the first example");
        int x = 5;
        int y = 6;
        int z = x + y;
        System.out.println("I just did a calculation");
        System.out.println("x + y = z");
    }
} // that's all, folks!
```

Program result

![Screenshot of program output](image-url)
Class definition

- The example program contains the definition of a class named Example 1
- The class definition begins with the **class heading**: public class Example1 {
- The class definition ends with an **end bracket**: }
- Any code that appears between brackets {} is called a **block of code**, the **body** of a class is a block

Method definition

- The body of the class consists of a single method
- The method begins with a method heading: public static void main (String[] args) {
- The body of the method, which is also a block of code, begins with the begin bracket and ends with a corresponding end bracket
- The end bracket for the method appears just before the end bracket for the class – the method is inside the class

Program instructions

- The body of the method consists of a set of instructions for the computer to perform
- There are three kinds of instructions in this program:
  - Method calls (messages)
  - Data declarations
  - Assignments
- Each instruction ends with the same symbol: a semicolon (;)
Method calls

- There are three examples of method calls in the program:
  System.out.println(“This is the first example”);
  ...
  System.out.println(“I just did a calculation”);
  System.out.println(“x + y + z = ” + z);
- A method call consists of:
  - The name of the calling object (System.out in this case)
  - The method name (println)
  - An argument list in parentheses

The println method

- System.out is the name of an object defined in Java’s standard library
  - Represents the standard output stream
  - On most systems, this is the screen
- The println method sends output to the stream object that calls it
  - Arguments to the method are the data to be output
  - Arguments can be quoted strings or values of simple data types

Arguments to println

- The arguments to println, like arguments to other methods, are expressions
  - An expression is a symbol or set of symbols that represents a value
  - Expressions take many forms – the simplest expression is a literal value
- String literal values in Java are any set of characters enclosed with double quotes (“”)
Arguments to println

- The + operator is used to **concatenate** strings with other strings or other expressions; for example:
  - System.out.println("" + x + " + y + " + z);
  - The entire contents of the parentheses is interpreted as a single string
  - The string concatenates the value of x with a plus sign, the value of y, an equals sign, and the value of z
  - Note that the plus sign in quotes is interpreted as a string, not as an operator

Data declaration statements

- In order to store values in memory, the programmer needs to set aside memory space to hold those values
- An instruction that allocates memory is a **data declaration** statement
- In the example program, there are three such statements, which set aside memory to hold three numbers

Data declaration statements

- The three spaces set aside for the numbers are called **variables**
  - Variables are memory spaces that hold single values
  - The value held in a variable can be changed many times during the course of a program
Data declaration statements

- The general syntax for a data declaration statement is:
  - `dataType identifier;` or
  - `dataType id1, id2, id3, ... idn;`
  - “`dataType`” is one of the standard Java data types described on the next slide
  - “`identifier`” (or “`id1`,” etc.) is a name chosen by the programmer to uniquely identify the memory space

Data types in Java

- Recall from the hardware lecture that different kinds of data values (whole numbers, real numbers, characters, etc.) require different amounts of memory
- Data types are reserved words in the Java language to indicate how much memory must be allocated to hold a particular named value
Assignment statements

- Once a variable has been declared, a value can be stored in the named memory space
- One way to store a value in a variable is via an assignment statement
- Syntax:
  Variable = expression;
  - “Variable” is a previously-declared variable
  - “expression” represents the value to be stored

Examples

- In the example program, there are three statements that combine data declaration and assignment:
  int x = 5;
  int y = 6;
  int z = x + y;
- Variables x and y are assigned literal integer values; variable z is assigned the sum of x and y

Comments

- There is one more kind of statement in the program, at the very end:
  // that’s all folks!
- This is an example of a comment
- Comments are a form of documentation, not code
- Unlike this silly example, comments are usually used to explain what is happening in the program
Comment types

- A single-line comment begins with two slash marks and ends at the end of the line; example:
  // this is a single-line comment
- A multi-line comment begins with a slash followed by an asterisk, and ends with an asterisk followed by a slash:
  /* this is a multi-line comment 
   this is part of the same comment 
   and this is the end of it */

Comments

- Comments are not code; they are ignored by the compiler and do not become part of the executable program
- Comments are useful when well-written; they can go a long way toward making source code maintainable