Programming Basics

15-110 Summer 2010

Margaret Reid-Miller

Java program structure

- Every Java program consists of one or more **classes**:
  - Each class contains **methods**:
    - Each method contains the **statements** (instructions) to be executed.

- The program starts its execution at the method called `main` and follows the instructions in the order specified.

Java Classes

- Java programs contain one or more *classes*, which are the basic units of code.

- The basic form of a Java class is at follows:

<> indicates that it needs to be filled in

```
public class <name> {
    <method>
    <method>
    …
    <method>
}
```

Indicates any number of methods

Methods

- A *__method__* is <u>named</u> sequence of instructions that performs some task or computation.
- Every Java program must have a method named <u>main</u> of the form

method header → `public static void main (String[] args) {`

```
        <statement>;

        <statement>;

            …

        <statement>;

    }
```

grouping characters

- The program execution starts at the `main` method.

Statements

- A *statement* is a single instruction for the computer to execute.

- Statements terminate with a semicolon (;) character.

- The statements in a method are executed in the order they appear.

- A statement can *call* or *invoke* another method; it requests that the computer executes the instructions of that method before proceeding to the following statement.

A simple program

```
/*
 * Prints three lines to the console.
 */
public class DreamDisplayer {

    public static void main(String[] args) {
        System.out.println("When I grow up ... ");
        System.out.println();  // blank line
        System.out.print("I want to be ");
        System.out.println("an astronaut.");
    }
}
```

Comment: Text to explain the code but is ignored by the compiler.

A *statement* that calls the method **print**

- **println** is a method already written for you.
- **System.out** is where to find the method.

```
OUTPUT:
When I grow up ...

I want to be an astronaut.
```

Strings

- A *string* is a sequence of characters that we "string" together.

    - In Java you can specify a string by writing the literal text inside a pair of double quotation (") marks.

- Examples of *string literals*:

    `"The following line is a one character string."`

    `"I"`

- A string cannot span more than one line:

    `"Not a valid`   WRONG!

    `String literal"`

Methods

```
public class MessageDisplayer{

    public static void main(String[]args){
        displayQuestion();
        System.out.println("DONUTS!");
        displayQuestion();
        System.out.println("BEER!");
    }

    public static void displayQuestion(){
        System.out.print("What does Homer like? ");
    }

}
```

method call

method definition

OUTPUT:
What does Homer like? DONUTS!
What does Homer like? BEER!

Method execution

Execute `main` method:

Execute `displayQuestion` method:

```
System.out.print("What does Homer like? ");
```
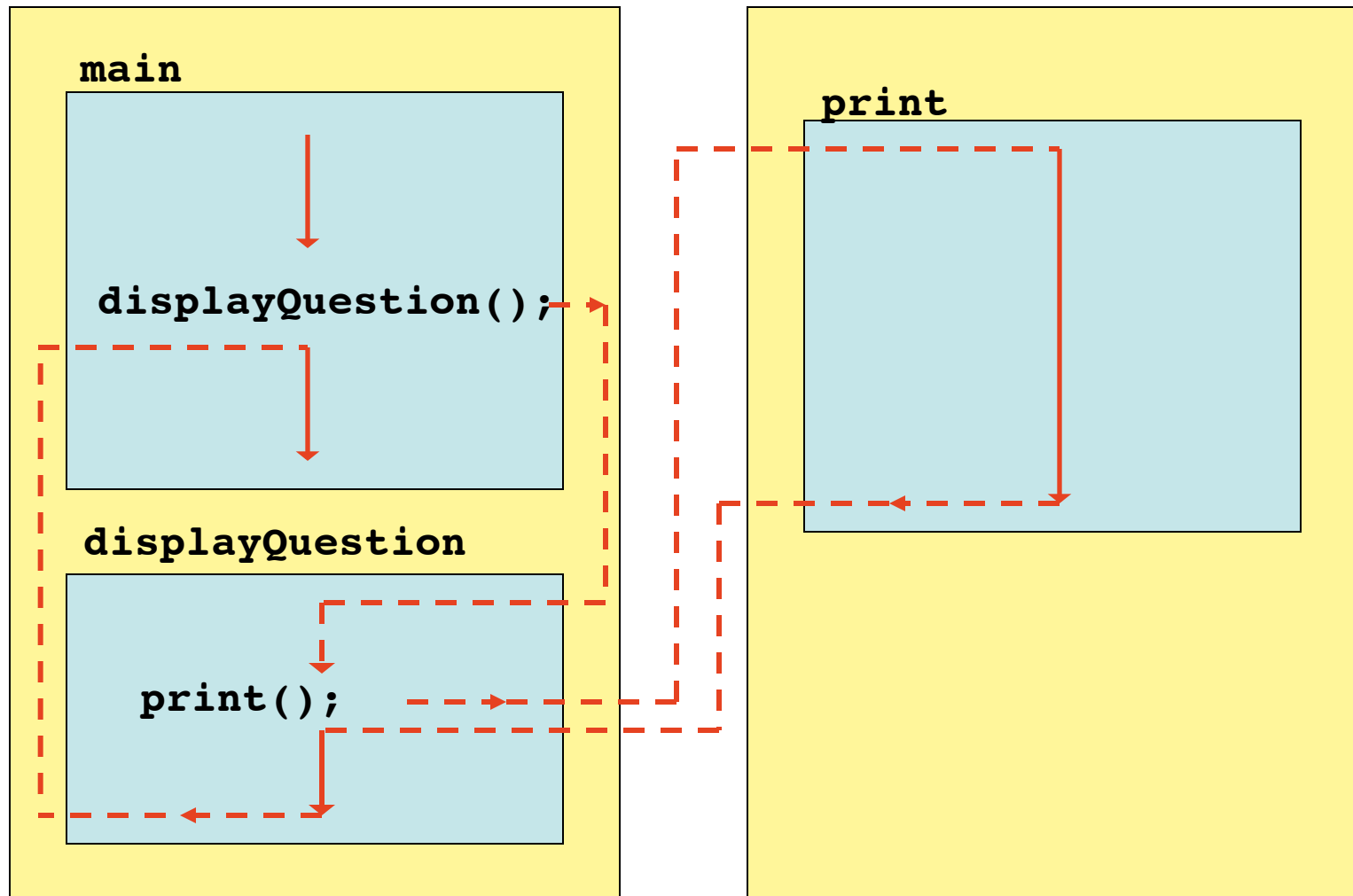
```
System.out.println("DONUTS!");
```

Execute `displayQuestion` method:

```
System.out.print("What does Homer like? ");
```

```
System.out.println("BEER!");
```

Method Flow of Control

main

displayQuestion();

**displayQuestion**

print();

print

Why Methods ?

- We use methods
  - to show the *structure* of a large program by decomposing it into smaller pieces and grouping related statements together in a method; and
  - to remove *redundancy* through reuse.

Methods can call methods

```
public class MethodCaller {

    public static void main(String[] args) {
        method1();
        method2();
        System.out.println("Bye");
    }

    public static void method1(){
        System.out.println("This is method1");
    }

    public static void method2(){
        System.out.println("Method2 calls method1");
        method1();
        System.out.println("Method2 is done");
    }
}
```

Output of MethodCaller

This is method1

Output from method1

Method2 calls method1

This is method1

Method2 is done

Output from method2

Bye
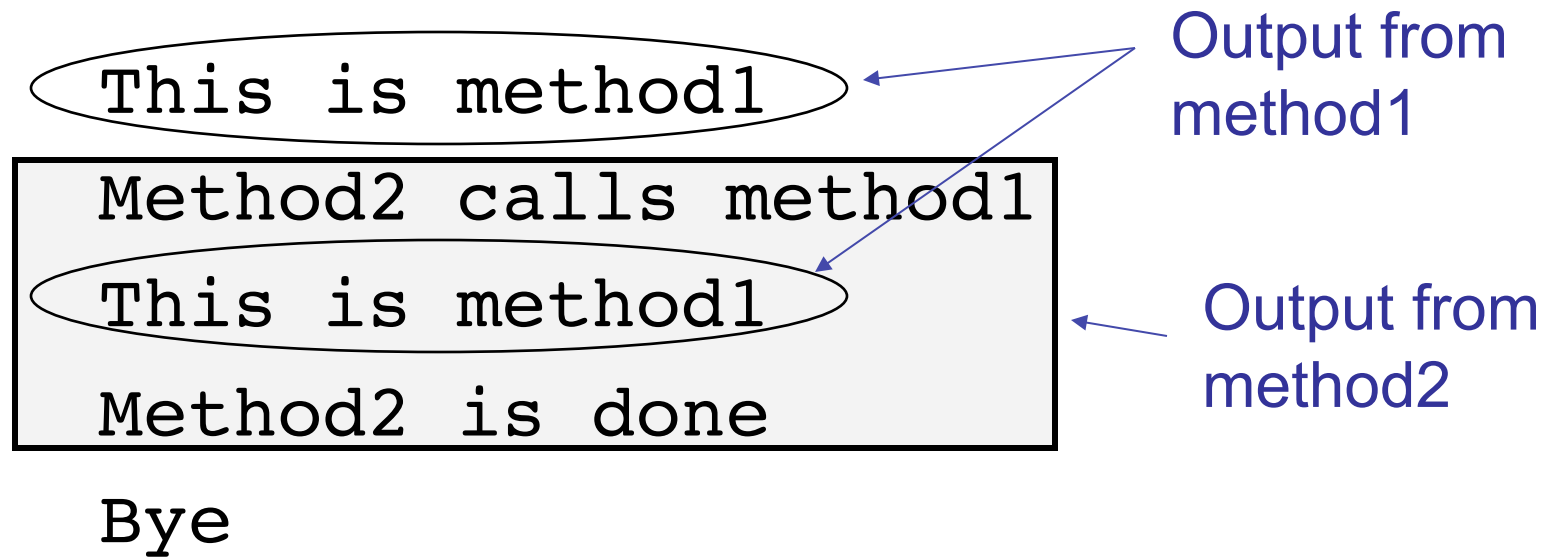
Exercise

- Write a program to print *banana* in block letters:

```
BBBB
B    B
BBBB
B    B
BBBB

 AAA
A    A
AAAAA
A    A
A    A

N     N
NN    N
N N N
N  NN
N     N
```

- Use static methods to reduce redundancy and to show the structure of the program.

Identifiers and Keywords

- *Identifiers* are names that specify different elements of a program such as class, method, or variable
  - can be any combination of letters, digits, _ or $
  - the first character must NOT be a digit
  - case-sensitive (`total` is different from `Total`)

  Examples: `main method1 maxCount`
  `        TUESDAY $amount Puzzle`

- *Keywords* are a set of predefined identifiers that are reserved for special uses.

  Examples: `public static void class`

Naming Conventions

- Java naming conventions help readers readily distinguish various Java elements:

  - **Class:** Starts with a capital letter
    - A class name should be a <u>noun</u> that describe an object type.

      e.g., `DreamDisplayer`, `Radio`

  - **Method:** Starts with a lower case letter
    - A method name should start with a <u>verb</u> and describe what the method does.

      e.g., `displayQuestion`, `getName`, `computeTax`

  - **Variable:** Starts with a lower case letter
    - A variable name should be a <u>noun</u> that describes what data it holds.

      e.g., `favoriteFood, name, taxRate`