Fundamentals of Data Structures

Mahesh Shirole

VJTI, Mumbai-19

Where to start ?

- Building data structures and algorithms requires that we communicate detailed instructions to a computer.
- An excellent way to perform such communication is using a high-level computer language.
- Conceptualizing a real-world problem using Object-Oriented Programming Language (OOPL), such as Java, C++, etc., is easy as compared to procedural language such as Basic, Cobol, C, etc.
- In OOPL, methods carry out a task, while data stores information.
- Example: The thermostat on a furnace, for example, carries out tasks (turning the furnace on and off) but also stores information (the current temperature and the desired temperature)

Objects in a Nutshell

- The idea of objects arose in the programming community as a solution to the problems with procedural languages
- An object contains both methods and variables
- A thermostat object, for example, would contain not only ***furnace_on()*** and ***furnace_off()*** methods, but also variables called ***currentTemp*** and ***desiredTemp***
- In Java, an object's variables such as these are called fields
- An object in a program correspond more closely to an object in the real world

Classes

- A class is a specification—a **blueprint**—for one or more objects
- An object would be enough for one programming instance
- You might want to make several objects of the same type (class)
- For example, you need several dozen thermostat objects in your program if you have to deal with furnace control programme of entire apartment building
- We choose **Java** as an OOPL in this course
- The Java keyword class introduces the class specification, followed by the name you want to give the class. Enclosed in curly brackets are the fields and methods that make up the class

class thermostat

```
class thermostat {
    private float currentTemp();
    private float desiredTemp();
    public void furnace_on() {
        // method body goes here }
    public void furnace_off() {
        // method body goes here }
} // end class thermostat
```

Creating Objects

- Specifying a class doesn't create any objects of that class
- To actually create objects in Java, you must use the keyword ***new***
- At the same time an object is created, you need to store a reference to it in a variable of suitable type. (A reference as a name for an object)

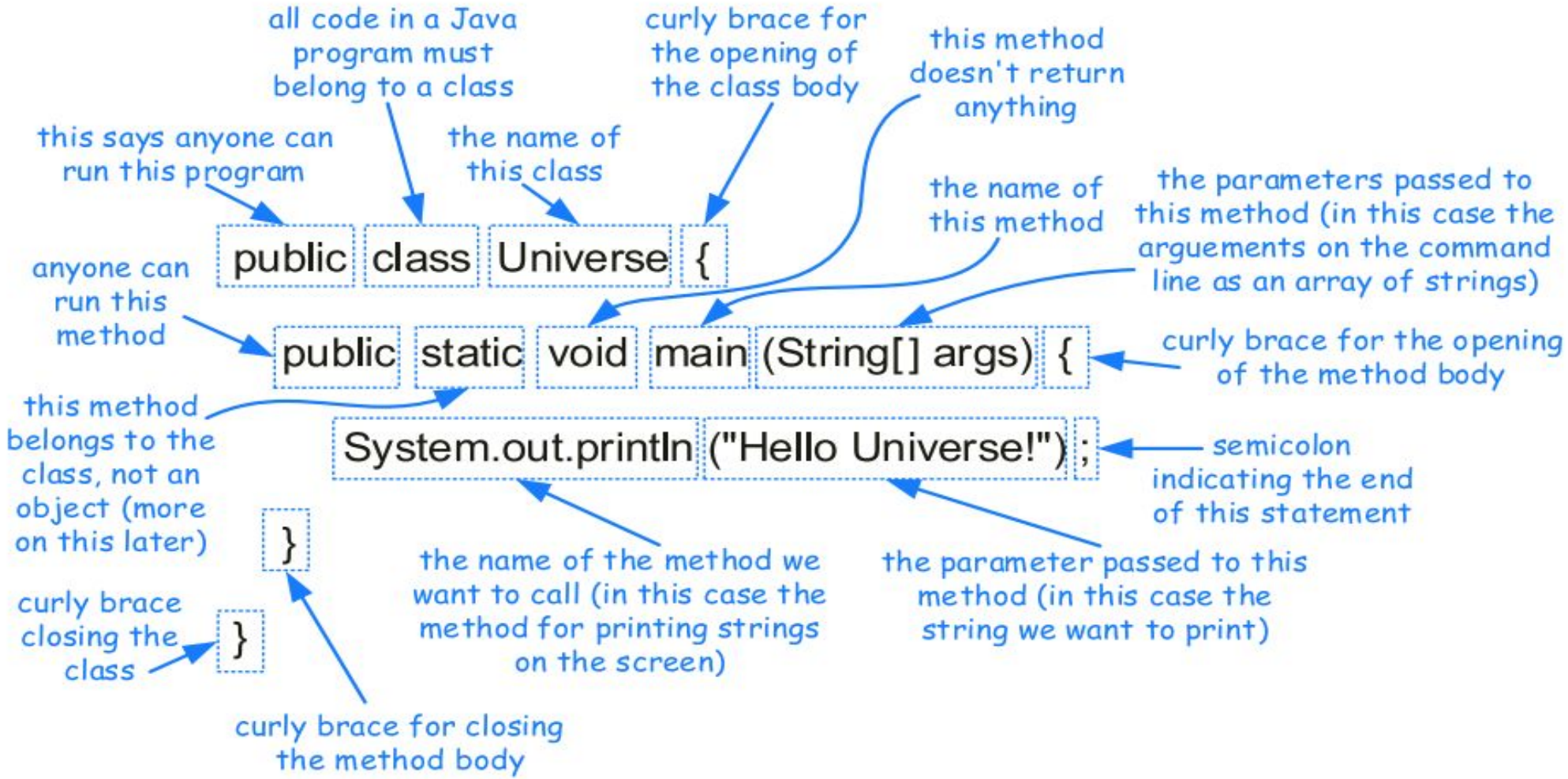
```
thermostat therm1, therm2; // create two references

therm1 = new thermostat(); // create two objects and
therm2 = new thermostat(); // store references to them
```

- Creating an object is also called instantiating it, and *an object is often referred to as an instance of a class*

How to interact with instances/objects?

- After you specify a class and create some objects of that class, other parts of your program need to interact with these objects
- Mostly we interact with object by using object's method rather than data
- Accessing Object Methods: To tell the therm2 object to turn on the furnace
  - **therm2.furnace_on();**
- The **dot operator (.)** associates an object with one of its methods

Java Program



What we learned?

- Objects contain both methods and fields (data)
- A class is a specification for any number of objects
- To create an object, you use the keyword *new* in conjunction with the class name
- To invoke a method for a particular object, you use the *dot operator*

Constructors

- Constructor is a special method that's called automatically whenever a new object is created
- A constructor always has exactly the same name as the class
- A constructor allows a new object to be initialized in a convenient way

Access Modifier

- The keywords public and private are access modifiers
- It determine which methods can access a method or field
- A field or method that is private can be accessed only by methods that are part of the same class
- A field or method that is public can be accessed by methods in other classes
- *Data fields* in a class are typically made **private** and *methods* are made **public**. This protects the data; it can't be accidentally modified by methods of other classes
- Any outside entity that needs to access data in a class must do so using a method of the same class

Inheritance

- Inheritance is the creation of one class, called the *extended* or *derived class*, from another class called the **base class**
- Inheritance enables you to easily add features to an existing class and is an important aid in the design of programs with many related classes
- Inheritance thus makes it easy to **reuse** classes for a slightly different purpose

Polymorphism

- Polymorphism involves *treating objects of different classes in the same way*
- For polymorphism to work, these different classes must be derived from the same base class

In practice, polymorphism usually involves a method call that actually *executes different methods for objects of different classes*