Classes and Objects

Chris La Pilla
CompSci 6
Summer 2007

Programming Languages

• Goal of programming language is to give us a framework in which we can solve problems
• Many different approaches to designing programming languages (procedural, functional, object-oriented, etc.)
• Regardless of the language, all code eventually becomes machine code so it can be executed
• So why do we have different programming languages?

Modularization

• Common goal of programming languages is modularization
• Instead of one giant list of instructions, break program up into smaller pieces (modules) with well-defined roles
• This way, it is easier to maintain and extend our software, because we can work on specific modules and ignore the rest of the code
• Different types of programming languages differ in the way they allow us to modularize programs

Object-orientation

• Idea: Consider some tasks:
  – calculating course grades, modeling a car, playing a board game, …
• Consider the "entities" (things) in the world that relate to these tasks
  – students, exams, grades, driver, engine, pedals, players, scores, dice, rules, …
• What is an abstract representation that's useful for all of these concepts?

Object-orientation

• Each entity has:
  – State: ("state of being") What are the current attributes or properties or configuration of the entity?
  – Behavior: What does the entity do? How do we interact with it?
• Object-oriented languages represent entities as objects, which have states (instance variables) and behaviors (methods)

Object-oriented program design

1. Decide the types of entities that are involved in our task
2. Define the state and behavior for each type of entity (classes)
3. Use specific instances of these entities (objects) in a program to solve the task
Classes

- A class defines all objects of a particular type
  - think of it as a blueprint
  - e.g. one String class, many String objects
- It describes the internal structure of all objects of that type:
  - instance variables: represent state – what attributes or properties do these objects have?
  - methods: define behaviors – how do we interact with these objects?

Objects

- An object is a particular instance of a class
  - e.g. every String object you create is an instance of the class String
- You can create many objects of the same class
  - e.g. `String string1 = “hello”;`
  - `String string2 = ”world”;
- Each object has its own state; that is, it has specific values assigned to its instance variables
- Methods of an object manipulate the state of that specific object

Example: computing grades

- See example code

State: Instance Variables

- Each object of a class has its own values for the instance variables
- In general, instance variables should be private and should only be accessed through methods
- This allows us to control exactly how these variables can be changed

State: Instance Variables

- Instance variables go inside class but outside methods
- Declaration contains:
  - Access specifier (public, private, protected)
  - Type
  - Identifier (by convention, should start with lowercase)
- Examples:
  - `private int totalscore;`
  - `private String name;`

Behavior: Methods

- Method declaration contains:
  - Access specifier (public, private, protected)
  - Return type (primitive type, class, or void)
  - Identifier (by convention, starts with lowercase)
  - Parameter list, enclosed in parentheses and separated by commas; each parameter has a type and an identifier
- Method body goes in curly braces {
- For method types other than void, last line must be a return statement which indicates the value that is passed back to the caller
Behavior: Methods

• Examples:

```java
//public void addGrade(int score){
    totalScore = totalScore + score;
    numScores +=
}
public double getGradeAverage(){
    return (double) totalScore / numScores;
}
```

Constructors

• A constructor is a special method that is responsible for initializing the instance variables of an object
• Constructors are declared like other methods, except they have no return type:

```java
public StudentRecord(String studentName){
    name = studentName;
}
```
• New objects are declared by using the `new` keyword and calling the constructor:

```java
StudentRecord record = new StudentRecord("Bob");
```

Access Specifiers

• public: variable/method can be accessed by any other class
• private: variable/method can be accessed only within methods of this class
• protected: related to inheritance, we’ll discuss this later in the course

Encapsulation

• The code inside a class is its implementation
• Once we’ve implemented a class (and tested that it works correctly), we no longer need to pay attention to these details
• Instead, we are only concerned with its interface, i.e. how do we use it?
  – What are its methods?
  – What information do we pass in as parameters?
  – What information is returned to us?
• We treat the objects of the class as "black boxes"