Classes and Objects in Java

Object-oriented programming

Outline

- Working with objects
- Declaring a class
- Attributes, methods, and access control
- Constructors
- Interface / Implementation

Readings:
- *Java How to Program*, Sections 3.1-5, 3.7, 8.5, 8.6, 8.15

Java program

- A Java program is a collection of objects. They do things (its methods) and asks other objects to do things (calling methods of others).
- Each object has its own state (data members / instant variables)
- Each object is instantiated from a class
- Each class is usually specified in one source file (file name is the same with class name)
- Every line of code you write in Java must be inside a class (not counting import directives)

```java
// GradeBook.java
public class GradeBook
{
    // display a welcome message to the GradeBook user
    public void displayMessage()
    {
        System.out.println("Welcome to the Grade Book! ");
    }
    // end class GradeBook
}
// GradeBookTest.java
public class GradeBookTest
{
    // main method begins program execution
    public static void main( String args[] )
    {
        // create a GradeBook object and assign it to myGradeBook
        GradeBook myGradeBook = new GradeBook();
        // call myGradeBook's displayMessage method
        myGradeBook.displayMessage();
    }
    // end class GradeBookTest
}
Welcome to the Grade Book!
```
Objects and Object references

Objects of pre-defined classes must be explicitly created by `new` operator.

Objects are manipulated via `references`.
- `Object references play the roles similar to pointers`.

```
public class GradeBookTest
{
    // main method begins program execution
    public static void main( String args[] )
    {
        // create a GradeBook object and assign it to myGradeBook
        GradeBook myGradeBook = new GradeBook();
        // call myGradeBook’s `displayMessage` method
        myGradeBook.displayMessage();
    }
} // end class GradeBookTest
```

```
public class GradeBook
{
    // display a welcome message to the GradeBook user
    public void displayMessage()
    {
        System.out.println( "Welcome to the Grade Book!" );
    }
} // end class GradeBook
```

How to Declare a Class

Attributes, methods, and access control

- Attributes:
  - `object's data`
- Methods:
  - `tasks that the objects can do`
- Access modifiers:
  - `public`: Accessible anywhere by anyone
  - `protected`: Accessible only to the class itself and to its subclasses or other classes in the same "package"
  - `private`: Only accessible within the current class

```
public class GradeBook
{
    private String courseName;

    public void getCourseName() {
        return courseName;
    }
}...
```
Classes and objects in Java

Invoking Methods

```java
public class GradeBook {
    public String getCourseName() {...}
    public void displayMessage()
    {
        System.out.print( "Welcome to the grade book of ");
        System.out.println( getCourseName() + "!");
        ...
    }
}
```

Methods of a class can be invoked by another method of the same object to work on the object at hand.

Invoking methods of other objects to ask them to things

```java
public class GradeBookTest {
    public static void main( String args[] )
    {
        GradeBook myGradeBook = new GradeBook();
        myGradeBook.displayMessage();
        ...
    }
}
```

Overloading methods

Methods of the same class can have the same name but different parameter lists.

```java
public class GradeBook {
    ...
    void displayMessage();
    void displayMessage("Good luck!");
    ...
}
```

```
//some client code
```
Constructors
Initialization of object’s data

- Every class has special “method”(s) called constructor to initialise objects’ data members
  - A constructor is invoked when an object is to be “created” / “allocated” by using “new” operator
    ```java
    GradeBook myGradeBook = new GradeBook();
    ```
- A class may have multiple constructors (overloading)
  - Distinguished at compile time by having different parameter lists
- When no constructors are defined in source code, a default constructor that requires no arguments and does nothing will be automatically provided.
  ```java
  public GradeBook() {
      courseName = "";
  }
  ```

Implementation vs. Interface

- GradeBookTest: a “client” of GradeBook
- Implementation
  - Data structures and code that implement the features (instant variables and methods)
  - Usually more involved and may have complex inner workings
  - Clients don’t need to know
- Interface
  - The controls exposed to the “client” by the implementation
  - The knobs on the black box

Encapsulation / information hiding

- “Don’t expose internal data structures!”
- Objects hold data and code
  - Neither is exposed to the end user or “client” modules.
- Objects expose an interface
  - A cat, its look vs. its internal organs
  - A TV, its screen & buttons vs. the stuff inside the box
- Complexity is hidden inside the object
  - Make life easier for clients
  - More modular approach
    - Implementation changes in one component doesn’t affect others
  - Less error-prone