Application Programming

Week 2

Lecture 1

Deliverables

- Java Classes in more detail

Discussion

- Getter
- Setter
- Constructor
- `toString()` method
- Note: We watched the following video from the coursera course Object Oriented Programming in Java
  - Core: Defining Classes and Creating Objects - Welcome and Project Overview: Visualizing Data | Coursera

A Java Class -- `Account.java`

- Create a class to represent an account. In our example, all accounts will have a name.
  - First, create a new file and declare the class.
  - Next, since all accounts have a name, create a class variable to hold the `name`.
    - What is the advantage to the name variable being `private`?
    - How can classes using Account objects access/modify name?
- Test out the class.

A Java Class -- `AccountTest.java`

- Create a class to test the `Account` class.
- First, create a new object, `Account`
- Prompt the user for a name.
- Read in the user input.
- Assign this input as the account `name`
- Print the name of the account.

A Java Class -- `Account.java`

- An improvement on the code would be to be able to set the name of the account when it's "opened".
  - Requires a **`constructor`** to be declared.
    - Called when the object is instantiated.
    - No return type.

A Java Class -- `Account.java`

- In reality, accounts will also have a balance.
- Add a balance variable to the code.
- Create getters and setters for the new variable.
- Update the constructor.

Java Programming Practices

- As a Java Developer, there are some key implementation rules you can follow to facilitate other developers using your code.
  - Style! (Eclipse will help with this)
  - Implement methods such as:
    - `main()`
      - provide an example of how your object would be utilized.
    - `toString()`
      - returns a String representation of your object.
    - `equals()`
      - provides a way for users to compare instances of your object to other instances.
      - This also gives you control over what is relevant to differentiate your objects.
    - getters & setters
      - methods to get and set the value of class variables.
      - Getters retrieve the value only.
      - Setters update the value.

Java Programming Practices

- As a Java Developer, there are some key implementation rules you can follow to facilitate other developers using your code.
  - Style! (Eclipse will help with this)
  - Implement methods such as:
    - `main()`
      - provide an example of how your object would be utilized.

```
public class Account
{
.
.
.
   public static void main( String[] args ){
    Account a = new Account( "John Deer", 700.75 );
    System.out.print( a.getName() + " " );
    System.out.println( a.getBalance() );
   }
}
```

Java Programming Practices

- As a Java Developer, there are some key implementation rules you can follow to facilitate other developers using your code.
  - Style! (Eclipse will help with this)
  - Implement methods such as:
    - `toString()`
      - returns a String representation of your object.

```
public String toString() {
    // String.format is a static method in the String class.
    // It is similar to sprintf in C.
    return String.format("Account object: name = %s,
                balance = $%.2f", name, balance);
}
```

      - To make use of this method, users can enter this statement:

```
Account myAccount = new Account( "Jane Doe", 150.00 );
System.out.println( myAccount );
```

Java Programming Practices

- As a Java Developer, there are some key implementation rules you can follow to facilitate other developers using your code.
  - Style! (Eclipse will help with this)
  - Implement methods such as:
    - `equals()`
      - provides a way for users to compare instances of your object to other instances.
      - This also gives you control over what is relevant to differentiate your objects.

```
public boolean equals( Account account2 ) {
    return this.getName().equals( account2.getName() );
}
```

Java Programming Practices

- As a Java Developer, there are some key implementation rules you can follow to facilitate other developers using your code.
  - Style! (Eclipse will help with this)
  - Implement methods such as:
    - getters & setters
      - methods to get and set the value of class variables.
      - Getters retrieve the value only.
      - Setters update the value.

```
public class HelloWorld {
      private String message;

      public String getMessage() {
            return this.message;
      }
      public void setMessage( String text ) {
            this.message = text;
      }
}
```

A Java Class -- `Account.java`

- In reality, accounts will also have a balance.
- Add a balance variable to the code.
- Create getters and setters for the new variable.
- Update the constructor.