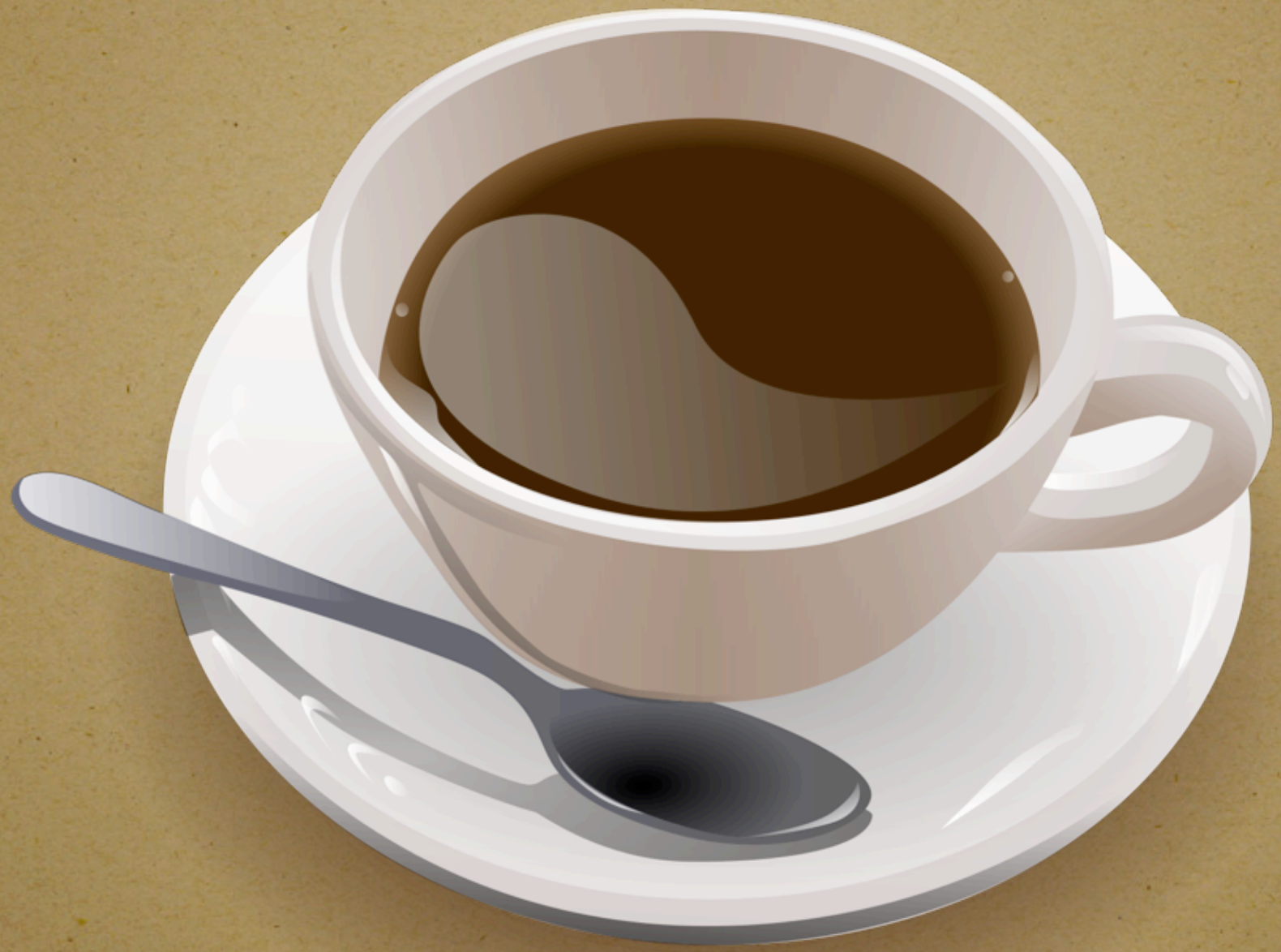
Image Credit: Julius Schorzman



Java Lesson 2

Concept Review

Abstract Classes

- Abstract classes are generic classes that aren't 100% complete. They're missing details that need to be filled in by a subclass.

```
public abstract class Worker
{
   protected double salary;

   // Get() method.
   public double getSalary()
   {
      return salary;
   }

   // Since we don't know what type of work this employee is doing,
   // we'll simply create an "abstract" method that the subclass
   // will have to provide.
   public abstract void setSalary();
}
```

- Notice that the setSalary() method isn't complete. Any subclass that extends this class will have to provide its own rendition of this method.

Abstract Classes

- To fill in the details, the subclass must "extend" the abstract class and fill in the remaining details for its incomplete, abstract methods.

```
public class Firefighter extends Worker
{
    // The subclass provides the setSalary() method's details.
    public void setSalary()
    {
        salary = 42500.00;
    }
}
```

- Notice that the Firefighter class provides the details for the abstract setSalary() method above.

Abstract Classes

- The best thing about abstract classes is that you can create a wide variety of subclasses that can use them. In this example, we'll create another employee, called the PostalWorker.

```
public class PostalWorker extends Worker
{
    // The subclass provides the setSalary() method's details.
    public void setSalary()
    {
        salary = 37300.00;
    }
}
```

- Notice that it also extends the Worker class and has to provide its own version of the setSalary() method.

Abstract Classes

- Since an abstract class **cannot** be instantiated as an object, you must instantiate its subclasses instead. Here's an example showing how to instantiate both the Firefighter and PostalWorker classes:

```
public static void main(String[] args)
{
   Firefighter bob = new Firefighter();
   bob.setSalary();

   PostalWorker dave = new PostalWorker();
   dave.setSalary();

   // Now, let's display the worker's salaries.
   System.out.println("Firefighter Bob earns: $" + bob.getSalary());
   System.out.println("PostalWorker Dave earns: $" + dave.getSalary());
}
```

- When the program is run, it results in the following:

```
Firefighter Bob earns: $42500.0
PostalWorker Dave earns: $37300.0
```