A Simple Java Program

```java
/**
 * A (very) slightly more interesting version of hello, world
 * @author David Mathias
 */

public class Dylan {
    public static void main(String[] args) {
        // Print the first stanza of 'Like a Rolling Stone' by Bob Dylan
        // Each System.out.println command prints a single line and
        // automatically puts a newline character at the end (sending
        // the cursor to the next line).
        System.out.println("Once upon a time,\n        
        You dressed so fine,\n        
        Threw the bums a dime,\n        
        In your prime, didn't you?\n        ");
    }
}
```

The Result of the Program

When we run the code:

1. main() method executes.
2. Each line in main is executed in turn

Looking at this code, and the output result, what do you know about what each line of code does?

An Object

- This object is everyday, and yet complicated
- **Interacts** with other objects
  - Cell towers
  - Other Bluetooth devices
  - You!
- **Made** of other objects
  - Most of us can figure out how to use it as a **black box**
  - That is, we can use it properly without knowing **exactly how** it does what it does
A Software Object

- Like "real" objects, these can also be "black boxes"
- A piece of code, but also:
  - Interacts with other objects
  - Made of other objects
  - Has a state (what it’s like now)
  - Has behavior (things done to it or things done by it)
- Often, we can figure out how to use it as a black box
  - We can use it properly and innovatively without knowing exactly how it works

Object Classes

- Every Java object belongs to a class (group)
  - Tells us what kind of object it actually is
  - The members in the group tells us something about the object's behavior and attributes
- If you are supplied with an existing class to use in your own code, you need to know how to use it properly
  - Once you know how the class works, you can use any individual object from that class

Class Diagrams

- Describes a class and how it can be used properly
  - Sketch of properties and behaviors of objects in the class
  - No extra details about how they actually work

Class Name
attributes (instance variables)
operations (methods)

<table>
<thead>
<tr>
<th>CellPhone</th>
</tr>
</thead>
<tbody>
<tr>
<td>True/False isOn</td>
</tr>
<tr>
<td>Number phoneNumber</td>
</tr>
<tr>
<td>List of #s addresses</td>
</tr>
<tr>
<td>turnOn</td>
</tr>
<tr>
<td>turnOff</td>
</tr>
<tr>
<td>answerCall</td>
</tr>
<tr>
<td>callNumber</td>
</tr>
<tr>
<td>checkVoiceMail</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Class Name
attributes (instance variables)
operations (methods)

<table>
<thead>
<tr>
<th>DrawingGizmo</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;constructor&gt;&gt;</td>
</tr>
<tr>
<td>DrawingGizmo()</td>
</tr>
<tr>
<td>&lt;&lt;update&gt;&gt;</td>
</tr>
<tr>
<td>void moveForward()</td>
</tr>
<tr>
<td>void turnClockwise()</td>
</tr>
<tr>
<td>void turnCounterclockwise()</td>
</tr>
<tr>
<td>void dontDraw()</td>
</tr>
<tr>
<td>void draw()</td>
</tr>
<tr>
<td>void delay2Sec()</td>
</tr>
</tbody>
</table>

Class Diagrams, cont’d.

- We want to know what a DrawingGizmo can do
  - Its class diagram tells us
  - Now we can use it!

Class Name
attributes (instance variables)
operations (methods)
Class Diagrams, cont’d.

- **No attributes!**
  - **DrawingGizmo** has no attributes given here, only methods for use
  - That’s fine: it’s just a simpler object than some

```java
<<constructor>>
DrawingGizmo()
<<update>>
void moveForward()
void turnClockwise()
void turnCounterclockwise()
void dontDraw()
void draw()
void delay2Sec()
```

Class Diagrams, cont’d.

- **Constructor**
  - **Constructor method**
    - Used to create an object
    - Has same name as class
    - Has no “return type” in front (unlike others that have `void`)

```java
<<constructor>>
DrawingGizmo()
<<update>>
void moveForward()
void turnClockwise()
void turnCounterclockwise()
void dontDraw()
void draw()
void delay2Sec()
```

Class Diagrams, cont’d.

- **Other methods**
  - **Update methods**
    - Things object can do
    - Usually change something about the object
    - Also called **mutators**
    - These return `void`, i.e., they have no output after they run (we’ll worry about this later on in the course)

```java
<<constructor>>
DrawingGizmo()
<<update>>
void moveForward()
void turnClockwise()
void turnCounterclockwise()
void dontDraw()
void draw()
void delay2Sec()
```

Creating a Variable in a Program

- To write a proper program, we must pay close attention to **syntax**
  - The rules of grammar, punctuation, spelling for Java

  Before we start using objects, we have to give them **names**, and our program must also tell the computer **what kinds of things** they are
  - Names must be **distinct** so the computer can tell which object is which
  - Spelling counts! An object named “pen” is not named “Pen”
  - **Meaningful** names are a good idea! In fact, they’re essential!

  Java syntax for naming a single object (variable declaration):
  ```java
  className objectName;
  ```
  - `className`: the kind of object it is
  - `objectName`: the name we want to give the object

  As an example, we declare a **DrawingGizmo** variable, “pencil”:
  ```java
  DrawingGizmo pencil;
  ```
Creating an Object in a Program, cont’d

- For complex Java objects, picking a variable name is not enough.
- To use a DrawingGizmo in a program, we must make sure we actually create such an object (instantiation).
- Java syntax for all object instantiation is essentially the same:
  `objectName = new constructorName();`
  - `objectName`: the variable name we want to use for the object.
  - `constructorName`: name of constructor method for the object (this is what builds the object; in Java, always the same as the class name itself).

Here, we instantiate DrawingGizmo object named “pencil”:
`pencil = new DrawingGizmo();`

Method Calls for Doing Things

- Once we have created the DrawingGizmo called pencil, we want to draw something with it.
- Things it can do are called methods.
- Use them by making method calls.
- Again, must follow Java syntax:
  `objectName.methodName();`
- `objectName`: name of object.
- `methodName`: name of method.
- The dot in the middle is a must, and so are the parentheses at the end!

For example, to make pencil move:
`pencil.moveForward();`

The Result of the Program

- When we run the code:
  1. `main()` method executes.
  2. `DrawingGizmo` object is created for use.
  3. Methods found in the `DrawingGizmo` class are executed to draw image.

Looking at this code, and the output result, what do you know about the `turnClockwise()` method?

Important Java Syntax I: Identifiers

- Each object in a Java program has an identifier (name).
- This includes:
  - any `class`, like HelloWorld or DrawingGizmo
  - any `variable`, like pen or pencil
  - any `method`, like moveForward()
  - anything else...
- Each identifier in a Java program:
  - will contain one or more alphanumeric characters: `a-z, A-Z, 0–9`
  - can also contain the `underscore` or `dollar` symbol: `_`, `$`
  - must not begin with a number (e.g., `a3` is legal, but `3a` is not)
- Other rules:
  - Names must be unique within same scope
    - This means that, inside any block defined by braces (`{...}`), we cannot have two different elements with the same identifier
  - Names are case sensitive
  - Names should be meaningful (think of “should” as “must”)

- 28 January 2020
- Software Design I (CS 120)
Every working Java program must define at least one class of object. For instance, we cannot call it DrawingGizmo, since that is the name that already belongs to another piece of code we are using in this program.

The format of the class definition we will use in most cases is:

```java
public class ClassName {...}
```

Every working Java program must begin with a special method called `main()` that always looks exactly the same on its first line. This method uses constructors like `DrawingGizmo()`, to actually create all the other objects we need and run their methods for us.

For This Week

- Meetings this week:
  - Tuesday, Friday: regular classroom
  - Wednesday: in the CS Lab (16 Wing)
    - You can bring your own computer if you like, but we will ensure everyone can log in to lab machines
    - If using your own machine, make sure to install Java and Eclipse before the lab (see class website, Resources section)

- Please obtain the online text ASAP
- First reading assignment due: 12:00 PM, Thursday 06 February

- Office Hours: Wing 212
  - Monday/Wednesday/Friday: 11:00 AM–12:00 PM
  - Tuesday: None this week (normally 3:15 PM–4:15 PM)