Today’s topic:

- Writing classes.
Die Objects

- A die has physical attributes:
  - a specific Number of Sides
  - colors (surface and pips)
  - Material
  - Dimensions
  - etc...

- I can complete certain actions with the die:
  - Count the sides
  - Pick it up
  - Roll it
  - Read the value on top
  - etc...

How to define a class?

- There are four key parts to defining a class.

  **Name** – I have to give it a name, so I know what to call the type of objects I create.

  **Instance Variables** – the data/attributes the object will maintain.

  **Constructors** – specialized methods that define how the object is initialized when instantiated.

  **Class Methods** – the actions or behaviors that this object can complete.
Giving it a Name

Syntax:

```
public class ClassName
```

For Example:

```
public class Die
public class DiePanel
```

---

Declaring Instance Variables

Syntax:

```
accessLevel dataType variableName;
```

For Example:

```
private int age;
private String name;
```

- All instance variables are declared private to support encapsulation.

- Weisfeld (page 65-66) refers to these as object attributes, these values are unique to a specific instance of a class.
Declaring Class Constants

Syntax:

```
accessLevel static dataType variableName;
```

For Example:

```
private static float PI;
private static int PIP_SIZE;
```

- Weisfeld (page 66-67) refers to these as class attributes, these values are shared by all instances of a class.

Constructors

- When an `new` instance of a class is created, the constructor is called and executed.
  - The constructor creates a new instance and allocates the required memory.
  - Then the actual constructor code is executed to complete any initialization that is required.
  - Typically the constructor is where the programmer sets any initial values of the instance variables.
Declaring Constructors

- Constructors are *specialized* class methods that have the *same name* as the class and initialize the instance variables.

Syntax:

```java
accessLevel className( inputParameters )
```

For Example:

```java
public Human()
public Human(String aName)
```

Typically, constructors are part of the *public interface* of the class, so we declare them *public*.

Default Constructor

- If the class does *not* have a constructor defined, a *default* constructor is provided by the compiler.

  - Therefore, all classes have *at least* one constructor.

  - It is a *good idea* to always write a constructor to *explicitly define* the initialization.

  - It is also a *good idea* to always *initialize* all of the instance variables as well.
Declaring Class Methods

Syntax:

```
accessLevel returnType methodName( inputParameters )
```

For Example:

```
public void add(int value1, int value2)
public String getName()
```

Any method that will be available to user of your class must be declared public.

The class may also contain private helper methods, to allow you to simplify your implementation.

---

Calling Methods

- To access an object’s method uses dot notation.
  - `objectName.methodName()`

- Parameters must match type and order of available methods.
Types of Methods

- **Accessor** methods – return values based on the current state of the object (read only).
  
  ```java
  Die.getNumSides();
  ```

- **Mutator** method – update instance variable values.
  
  ```java
  Die.setNumSides(6);
  ```

Types of Methods

- Some methods may change the state of the object as a **side effect**.
  
  ```java
  Die.roll();
  ```

- Some methods **return** values, some **do not**. The method declaration tells you. `(void OR String, Die, etc.)`
Local Variables

- Implementing a *class method* often requires keeping track of *additional* values.

- Weisfield (*page 64*) refers to these as *local attributes*.

- These variables are initialized and used during the *execution* of the method, but are *not* retained or available to other parts of code.

Method Overloading

- You can use a method name *more than once* (including constructors).

- The *overloaded method*, must have different method *signatures* to differentiate them.

  ```java
  public int add(int value1, int value2)
  public float add(float value1, float value2)
  ```