Classes & Objects

Object-Oriented Programming (OOP)

- OOP is a programming paradigm organized around objects rather than actions.
- These objects interact with each other
- Objects have attributes (member variables, fields)
- Objects have actions (member methods)

What is a class?

- Class is a template of an object, and we can create instances of that object based on that class
- Think of a class as a blueprint to what an object is

What is an object?

- An object is an instance of a class
- Objects are created from templates known as classes
- A class does not become an object until it is instantiated

Class & Object Example

- Consider an object in the real world. A soda can is a good example of this.
- The can has attributes that describe it. These are the can’s variables.
  - Color, contents, status of the can (open/close)
- The can has actions it can do. These are the can’s methods.
  - Open the can, pour contents out of the can, crush the can

Member variables

- Member variables (fields) are the variables defined inside a class.
- Think of member variables as the attributes of an object
  - Car is the object. Color and engine size are member variables of that object
- Member variables also keep track of an object's status
- Local variable: variables defined inside scope of a function

Member variables

- In our example of a can, these are what the member variables of the Can class will look like:

```
public class Can
{
    string color;
    bool isOpen;
    int percentageFull;
}
```

Member methods

- Just like member variables, we can create as many methods for our class as we want.
- The methods inside the class, or member methods, give functionality to the object.
- These methods are the actions of the class.

Member methods

```
public class Can
{
    public string color;
    public bool isOpen;
    public int percentageFull;

    0 references
    public void Open()
    {
        isOpen = true;
    }
    0 references
    public void TakeSip()
    {
        percentageFull -= 10;
    }
}
```

Methods added

Constructor

- A constructor is a special type of method.
- The constructor is called every time a new instance of a class is created
- We know a method is a constructor because it will have the same name as the class and it does not have a return type.

Constructor example

```
public class Can
{
    //member variables
    public string color;
    public bool isOpen;
    public int percentageFull;

    //constructor
    0 references
    public Can(string color, bool isOpen, int percentageFull)
    {
        this.color = color;
        this.isOpen = isOpen;
        this.percentageFull = percentageFull;
    }

    //member methods
    0 references
    public void Open()
    {
        isOpen = true;
    }
    0 references
    public void TakeSip()
    {
        percentageFull -= 10;
    }
}
```

Constructor added

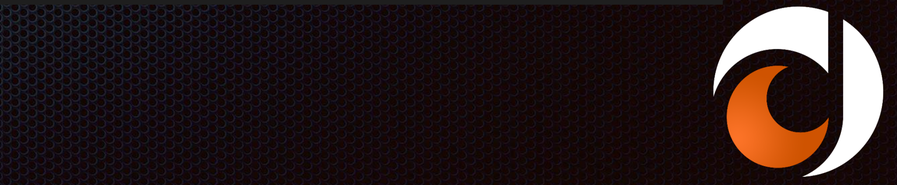
Constructor

- We can pass values into the constructor just as we can with a method
- These values typically are set equal to the member variables of the class
- This allows for each object that is instantiated from the same class to have unique member variable values, if desired

Instantiation

- Instantiation is creating an instance of a class (creating an object)
- To create an instance of a class (creating an object), we must use the “new” keyword.
- After creating an instance, we use that object to access the class variables and methods

```
Can grapeSoda = new Can("purple", false, 100);
Can mountainDew = new Can("green", false, 100);
```



Dot notation (.)

- To access an object's variables and methods, we use dot notation.

```
Can grapeSoda = new Can("purple", false, 100);
Can mountainDew = new Can("green", false, 100);
grapeSoda.color = "pink";
mountainDew.color = "yellow-green";
mountainDew.Open();
mountainDew.TakeSip();
```

Review Questions

- What is a class? What is an object?
- What is instantiation?
- What are member variables? Member methods?
- What is a constructor?
- What is dot notation?

Assignment

- Create a ClockRadio class
- Your clock radio should keep track of the time, current state of the alarm (on/off), and current radio station
- Your clock radio should be able to set an alarm and change to a new radio station
- Add as many other member variables and member methods as appropriate