What have we done so far?

- Background image displayed.
- UFO moving across the screen.
- Keyboard input is rotating the cannon.
Game Object

- ufo, cannon, cannonball game objects in our game.

- What do each of our game objects have in common?
  - Texture2D to store the image.
  - Vector2 to store the position.
  - UFO (and bullet when we add it) has velocity, and alive.
GameObject class

- Instead of storing all of these variables in our main Game1 class.
  - Create a new class: GameObject.
What is a class?

- It’s like a blueprint to create similar objects.
  - Let’s say we have a bicycle.
  - We want to store its speed, gear, and cadence (revolutions/minute, how fast we’re pedaling).
  - We also want to be able to change these values and we can do that with methods.
class Bicycle {
    private int cadence;
    private int speed;
    private int gear;

    Bicycle() {
        cadence = 0;
        speed = 0;
        gear = 1;
    }

    public void changeCadence(int newValue) {
        cadence = newValue;
    }

    public void changeGear(int newValue) {
        gear = newValue;
    }

    public void speedUp(int increment) {
        speed = speed + increment;
    }
}
Create a bicycle object

Bicycle myBicycle = new Bicycle();

myBicycle.changeCadence(100);
myBicycle.changeGear(5);
myBicycle.speedUp(10);
public attributes

- We have public attributes, that means we will be able to get and set those values outside of the GameObject class.
  - Let's say we declare a GameObject:
    
    ```
    GameObject enemy = new GameObject(enemyTexture);
    ```

- Since its attributes are public we can get or set them using "."
  
  ```
  enemy.position = new Vector2(800.0f, 200.0f);
  ```
**GameObject constructor**

- The constructor is called when we create a new GameObject.
  - Pass in as input, the texture for that object. (If it’s a UFO, we pass in a UFO texture)
- Then we need to initialize the values.

```java
public GameObject(Texture2D loadedTexture) {
    sprite = loadedTexture;
    center = new Vector2(sprite.width/2, sprite.height/2);
    position = new Vector2(0.0f, 0.0f);
    velocity = new Vector2(0.0f, 0.0f);
    rotation = 0;
    alive = false;
}
```
References

- At the top of each class you need the references to the XNA components we will be using.
- We can use the same references that we had at the top of Game1.cs:

```csharp
using System;
using System.Collections.Generic;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Storage;
```
Modify your code

- Create a GameObject class.
- Create a GameObject enemy and replace the code you used for the enemy with the GameObject enemy.