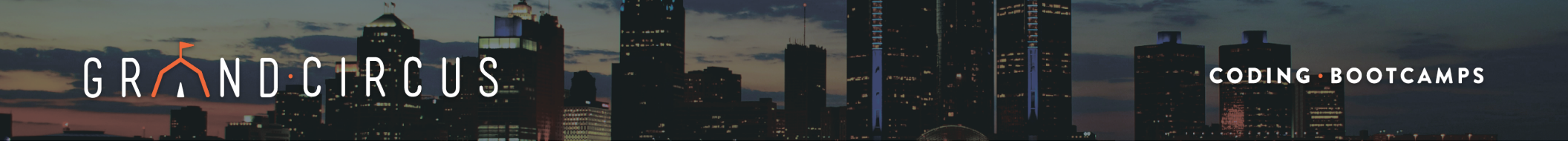

C# .NET BOOTCAMP

OBJECT ORIENTED PROGRAMMING

OBJECT ORIENTED PROGRAMMING

Explain some of the foundational concepts that we've covered already. How do they interact?

OBJECT ORIENTED CONCEPTS

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

OBJECT ORIENTED CONCEPTS

ABSTRACTION

- Hide the details and only show functionality.
- Example: When you do a phone call, you don't need to know the internal process of how the wires carry your voice.

OBJECT ORIENTED CONCEPTS

ENCAPSULATION

- Wrapping related functionality and data together as one unit.
- Example: Using classes to define an employee.

OBJECT ORIENTED CONCEPTS

INHERITANCE

- A class acquiring properties and behaviors of parent class.
- Provides code reusability, and used to achieve the forth concept, Polymorphism

OBJECT ORIENTED CONCEPTS

POLYMORPHISM

- Same task can be done in different ways depending on the object's type.
- Polymorphism is also the ability of an object to take on many forms. Example: An Employee is a Person, A Dog is an Animal

OBJECT ORIENTED CONCEPTS

POLYMORPHISM

- Same task can be done in different ways depending on the object's type.
- Polymorphism is also the ability of an object to take on many forms. Example: An Employee is a Person, A Dog is an Animal

CLASS VS. OBJECTS

- A class is a template or a blue print that describes the supported behavior and state of all objects of its type.
- An object is an instance of a class.

INSTANCE VARIABLES

An instance variable may be a value data type, an object created from a class such as the String class, or an object created from a user-defined class such as the Product class.

Created when the object is created, in contrast to static variables.

INSTANCE VARIABLES

To prevent other classes from accessing instance variable, use the private keyword to declare the as private.

We can declare the instance variable for a class anywhere outside the constructors and methods of the class.

SYNTAX

```
public|private|protected Type variableName;
```

EXAMPLES

```
public double price;
private int quantity;
private String code;
private Product product;
```

EXAMPLE IN CLASS

```
public class Product
{
//common to code instance variables here
private String code;
private String description;
private double price;
//the constructors and methods of the class …
//also possible to code instance variables here
private int test;
}
```

CONSTRUCTORS

WHAT ARE CONSTRUCTORS?

- A constructor is a method that is used to initialize the state of an object.
- A constructor must use the same name and capitalization as the name of the class.

CONSTRUCTORS

WHAT ARE CONSTRUCTORS?

- In the absence of a constructor C# will create a default constructor that initializes all numeric types to 0, all boolean types to false, and all objects to null.
- The parameter list of the constructor forms the signature of the constructor. Each constructor must have a unique signature.

SYNTAX

```
public ClassName([parameterList])
{
//the statements of the constructor
}
```

EXAMPLES

```
public Product(String code, String description, double price)
{
Mycode = code;
Mydescription = description;
Myprice = price;
}
```

THE THIS KEYWORD

Can be used to refer to instance methods and data defined inside the class.

Since C# implicitly uses the this keyword for instance variables and methods, we don't need to explicitly code it unless a parameter has the same name as an instance variable.

EXAMPLE

```
public void setX(int x)
{
this.x=x;
}
```

RECAP

What you should know at this point:

- What is OOP, and what are the concepts of OOP
- What are classes and objects
- Difference between classes and objects
- How to create classes and objects
- Know how to define instance fields
- Define and use constructors
- Using this keyword