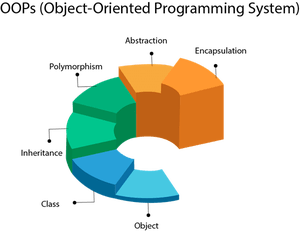
Object Oriented Programming Concept Questions (See Power Point Slides)

As you should know by now, there are 4 pillars of Object Oriented Programming.

Object Oriented Programming:

**A programming paradigm that leverages classes and objects and utilizes four programming principles (pillars): encapsulation, inheritance, abstraction, and polymorphism.**



picture credit: https://francescolelli.info/

**Encapsulation** is the idea that we want to keep data private. We use the principle of encapsulation so that users will be unable to access and modify the code. By using encapsulation, we can provide security to the application and prevent misuse.

The access modifier private declares a field or member variable as private and will only be able to be accessed within the class, unless public getter methods (a getter is a method that reads the value of a variable) are present in the class.

Example from program: This makes it so that no one can mess with the size of the GUI window.

From the GameDriver class:

//window size variables:

privare final int gameWindowWidth = 1000; //private encapsulates this variable so it is only available in the GameDriver class

private final int gameWindowHeight = 800; //final prevents the field from being changed

private final Dimension screenSize = new Dimension (gameWindowWidth, gameWindowHeight); //creates a new instance of Dimension called screenSize with gameWindowWidth and gameWindowHeight as its parameters. Dimension is a class from java.awt.

**Inheritance** is the concept that we can take the functionality that is available in an already existing class and extend it to a new class. This allows us to reuse code, thereby reducing the amount of code needing to be written to achieve a desired outcome.

This is a very powerful concept in OOP because it allows us to access the properties of an existing class and implement them quickly and easily.

Example from program: This uses Jframe functionality to build the window of the GUI.

Public class Gamedriver extends JFrame {

From the GameDriver class:

Code that initializes JFrame:

this.setTitle(“TEKcamp Pong!”);

this.setSize(screenSize);

this.setResizable(false);

this.setVisible(true);

this.setBackground(Color.CYAN);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

**Abstraction** is another useful concept in OOP. When we use abstracting we are achieving three useful outcomes: simplifying, generalizing, and conceptualizing. By using a google map example:

By simplifying we are hiding the details. We don’t show everything on the map. By generalizing we are creating a class that can be used over and over again. In our map example, blue lines represent rivers. We can use the concept of blue lines for rivers over and over again. We can conceptualize a location on the map by placing a red dot in its location. Obviously the red dot is a representation, not reality.

Example:

In this example, Bike is an abstract class that contains only one abstract method ride. Its implementation is provided by the Trek class.

 abstract class Bike{

  abstract void ride();

}

class Trek extends Bike{

void run(){System.out.println("riding safely");}

public static void main(String args[]){

 Bike obj = new Trek();

 obj.run();

}

}

**Polymorphism** is a useful concept in that it allows us to reuse code – we can reuse methods of an existing class when we create a new class. When a method in a subclass has the same name, same parameters or signature, and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to *override* the method in the super-class.

Example from Program: This is the method required for the Runnable interface that is being implemented by the Ball class.

@Override

public void run() { //method from the Thread class

try{

while(true) { //always true: infinite loop – keeps game going by continually calling this method

move();

Thread.sleep(50); //50 milliseconds to allow for computer multitasking

}

} catch (exception e) {

System.err.println(e.getMessage()); when an exception occurs, prints error message to console

}

}

}