the 4 object oriented principles: encapsulation, abstraction, inheritance, and  polymorphism

**Encapsulation Slide Set 12**

Wrapping related functionality and data together as one unit.

**Example:** Using classes to deﬁne an employee.

**Abstraction Slide Set 12**

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

**Inheritance Slide Set 12**

A class acquiring properties and behaviors of parent class.

Provides code reusability, and used to achieve the forth concept, Polymorphism.

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.

**Examples:** An Employee is a Person A Circle is a Shape

**Classes and Objects Slide Set 12 Page 12**

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.

**Constructors: Slides 12 and page 23**

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.

Additional Note about Constructors:

In the absence of a constructor Java will create a default constructor that initializes all numeric types to 0, all boolean types to false, and all objects to null.

The name of the class combined with the parameter list forms the signature of the constructor. Each constructor must have a unique signature.

Example:

public ClassName([parameterList])

{

//the statements of the constructor

}

public Product(String code, String description, double price)

{

Mycode = code;

Mydescription = description;

Myprice = price; }

}

**Getters and Setters**

They are also widely known as **setter** methods. Often a **setter** is accompanied by a **getter** (also known as an accessor), which returns the value of the private member variable. The mutator method is most often used in **object-oriented** programming, in keeping with the principle of encapsulation.

**abstract methods – Slides 14 page 5** An abstract method is also created using the abstract keyword. Abstract methods have no body. They cannot have private access. An abstract class does not have to contain abstract methods, but any class that does contain abstract methods must be declared as abstract.

Abstract Class - An abstract class does not have to contain abstract methods, but any class that does contain abstract methods must be declared as abstract. An abstract class can be inherited by other classes, but not used to create an object.

Example slide 14 page 6

**Interfaces** -Slides 14 page 8 - An interface is a special type of coding element that provides many of the advantages of multiple inheritance. An interface deﬁnes a set of public methods that can be implemented by a class. The code for an interface uses the interface keyword instead of the class keyword and contains only abstract methods. A class that implements an interface must provide an implementation for each method deﬁned by the interface.

**Access Modifiers: Slides 13 Page 10**

**Public** - Available to classes in all packages

**Protected** - Available to in the same package and to subclasses outside the package.

**package private (default)** - Only accessible within the package

**Private** - Available within the current class

**static methods and variables vs. instance methods and variables – Slide 12 Page 17** Created when the object is created, in contrast to static variables.

**Final Classes Methods and Variables** – Slide 13 Page 19 - We can prevent a class from being inherited by using the ﬁnal keyword. A ﬁnal method cannot be overridden by a subclass. All methods within ﬁnal classes are ﬁnal methods. The ﬁnal keyword may also be used to prevent a method from assigning a new value to a parameter.

Examples of Final:

public final class Book extends Product {

// all methods in the class are automatically final

}

public final String getVersion() {

return version;

}

public void setVersion(final String version) {

// version = "new value"; // not allowed

this.version = version;

}

**Extends Keyword – Slide 13 Page 11**

Declaring a subclass

public class SubclassName extends SuperclassName { }

**The This Keyword** – Slide 12 Page 27

Can be used to refer to instance methods and data deﬁned inside the class. Since Java 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. If you use the this keyword to call another constructor, the statement must be the ﬁrst statement in the constructor.

Syntax:

this.variableName

//refers to an instance variable of the current object

Example:

public void setX(int x)

{

this.x=x

}

**Super Keyword** – Slide 13 Page 12

We can use the super keyword to call a constructor or method of the superclass.

**Enums** – Could not find in class, we took a note on this in class - which is short for enumeration and it is a type that defines a list of options.

**public** **enum** Roshambo {

ROCK, PAPER, SCISSORS

}

null and NullPointerException  - Online from geeks for geeks - **NullPointerException is** thrown when program attempts to use an object reference that **has** the null value. These **can** be: Invoking a method from a null object. Accessing or modifying a null object's field.