

# CIS 226 – Intro to Java

## Object Inheritance

---

---

---

---

---

---

---

---

## Assignment 5 Tips for Assignment 6

- Methods and classes required JavaDoc comments

```
/**  
 * This is method validates input.  
 * @param input the input to validate  
 * @return returns true if valid, otherwise false  
 */  
public boolean validateInput(String input) {  
  
}
```

Look at javadoc comments and javadoc tags

---

---

---

---

---

---

---

---

## Fair Warning

- All code should:

- Follow a style

```
if(...) {  
    statement;  
}  
or  
if(...) {  
    statement;  
} else {  
    statement;  
}
```

- be properly indented
- methods should have descriptive names
- methods should implement a specific feature or task, not multiple tasks.
- Proper javadoc comments, with javadoc tags

---

---

---

---

---

---

---

---

## OOP - Inheritance

- ◆ A key feature of OOP
- ◆ Form of code reuse – Why?
- ◆ Allows the programmer to define a very general class and later define a more specific class by adding new details to the general class.

---

---

---

---

---

---

---

---

## The Base Class

- ◆ Suppose we build a class that contains information about people, a Person class.
- ◆ This class contains general information about a person: name, age, address, etc.
- ◆ We can use this class by adding new instance variables and methods to it which will contain the person's information.
- ◆ This general Person class is consider our Base Class because we reuse all of its variables and methods to build new classes upon.
- ◆ Base classes in java are also known as Super classes.

---

---

---

---

---

---

---

---

## The Derived Class

- ◆ Suppose we build a class that contains information about students, a Student class.
- ◆ We can reuse all the code from the Person class and add to it the new methods and instance variable for our student data.
- ◆ New instance variables might be, studentID, status, current gpa.
- ◆ This general Person class is consider our Base Class or Super class because we reuse all of its variables and methods to build new classes upon.
- ◆ The Student class is a derived class from the Person class because it uses all of the instance variables and methods from Person.

---

---

---

---

---

---

---

---

## The Derived Class

- ◆ Additional classes can be built upon the Person class. These classes are known as derived or subclasses of the Person class.
- ◆ We can have an Employee class that is derived from Person and we can also have classes built upon the Employee class. Classes such as Faculty, and Administrator classes.
- ◆ As well, we could build upon the Student classes by creating classes PartTimeStudent and FullTimeStudent that are subclasses of Student, which is a subclass of Person.
- ◆ Inheritance allows us to create new classes that are very specific from general class definitions.

---

---

---

---

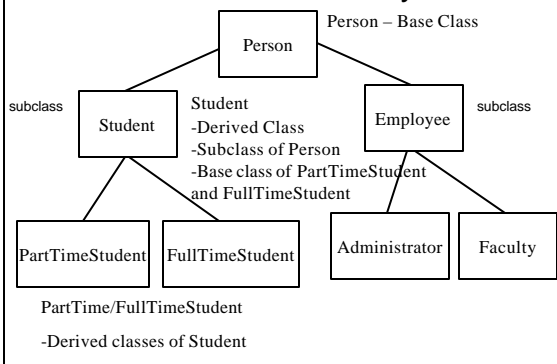
---

---

---

---

## Person Hierarchy



---

---

---

---

---

---

---

---

## Let's see the code

- ◆ This slide will refer to the online code.

---

---

---

---

---

---

---

---

## Access Modifiers

- A superclass's *public* members are accessible anywhere there is a reference to the superclass or one of its subclasses.
- A superclass's *protected* members may be accessed by the superclass's methods and by any subclasses

---

---

---

---

---

---

---

---

## How do we inherit?

- Java uses the keyword "extends" to implement inheritance.
- Unlike C++, Java does not support multiple inheritance. Thus, derived classes can only extend one base class.
- We have been using inheritance since the very beginning of class. When we use the Applet or JApplet class, we extend these classes. Thus, our classes inherit all of the JApplet methods and instance variables.

```
public class ClassName extends SuperClassName {  
or  
public class KingWishApplet extends Applet {  
...
```

---

---

---

---

---

---

---

---

## Inheritance with the Applet Class

- From the previous slide, if we extend the Applet class, we are given the `init()`, `start()`, and `paint()` methods from the base class, *Applet*.
- However, we **override** the methods in the Applet class to do specific functions to solve the problems given to us.
- Remember, since Applet is a base class it defines general functionality. This will not solve specific problems.

---

---

---

---

---

---

---

---

## Overriding Methods

- Overridden methods are methods with the same name, exactly the same parameter list, and return type as defined in the base class.
- Parameters must be the same number of parameters and types in each class, base and derived.
- This new definition of the method will replace the old definition in the class' super class. Thus, if we create a method with the following definition:

From the Person Class

```
public String toString() {  
    return this.fullName;  
}
```

Then in the derived class, Student, define the method "toString" that takes no parameters, we have overridden the base class' "toString" method.

```
public String toString() {  
    return super.toString() + " " + this.studentID;  
}
```

- Note: superclass methods can be accessed by using the keyword "super" to distinguish between the overridden method and the superclass' method.

---

---

---

---

---

---

---

---

## Overridden versus Overloading

- Overriding a method
  - Methods have the same name, number of parameters and type of parameters.
  - One method is defined in a base class and the other, overriding method, is defined in a derived class
  - Both methods have the same signature
  - When overriding methods access cannot be more restrictive in the subclass' method. It has to be the same or have more access.
- Overloading a method
  - Methods have the same name, with different parameters
  - Methods may or may not have the same return type.

---

---

---

---

---

---

---

---

## "Is a" relationship

- Inheritance results in a "is a" relationship.
- Subclasses assume all the *attributes* and *behaviors* from their Superclass
- When constructing new objects we should examine this in real life situations.
- if( subclass instanceof superclass)
  - Results in a true value
- Ex.
  - A Student is a Person
  - An Employee is a Person

---

---

---

---

---

---

---

---

## “Has a” relationship

- “Has a” results from a composition relationship
- Promotes objects within objects
- Student could have a reference to a Person object
- As a result, none of the Person’s attributes or behaviors will be inherited in the Student class

---

---

---

---

---

---

---

## Review

- Encapsulation – data hiding
  - Access with accessor methods - *getters*
  - Modification with mutator methods – *setters*
- Code Reuse via “*extends*”
- Superclass = Base Class
- Subclass = Derived Class
- “Is a” relationship denotes inheritance
- “Has a” relationship denotes composition
- Method overloading
- Method overriding

---

---

---

---

---

---

---

## Info

- Assignment 6 and 7 next week
- Quiz 4 today
- Performance Test 2 - Thursday

---

---

---

---

---

---

---