CS221: Inheritance, Polymorphism, Abstract classes

Resources:

**Sanders & Van Dam**: Chapter 5 (pp 1 57 – 163, 164 – 165.)

<http://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html>

<http://math.hws.edu/eck/cs124/javanotes6/c5/s5.html>

(Section 5.5: Inheritance, Polymorphism, and Abstract Classes)

* **Polymorphism** is implemented using a technique called ***late*** (or ***dynamic***) ***method binding***: which exact method to call is determined **at run time**.
* Polymorphism ensures that the appropriate method is called for an object of a specific type when the object is **declared a**s a more generic type
* Different from **overloading;** overloading is resolved by the **compiler (*early binding*)**

**Benefits of Polymorphism**

* **Polymorphism** enables programmers to deal in **generalities** and let the execution-time environment handle the specifics.
* Programmers can command objects to behave in manners appropriate to those objects, without knowing the types of the objects (as long as the objects belong to the same inheritance hierarchy).
* **Polymorphism** promotes **extensibility**: Software that invokes polymorphic behavior is independent of the object types to which messages are sent.
* New object types that can respond to existing method calls can be incorporated into a system without requiring modification of the base system. Only client code that instantiates new objects must be modified to accommodate new types.

**Exercises:**

1) Explain the difference between the following: (Resource: <http://math.hws.edu/eck/cs124/javanotes6/glossary.html>)

1. The actual type and the declared type of a variable?

The actual type implements the declared type that means that the actual type is a subclass of the declare type. In short the declare type is the nature of the variable while the actual type is where the variable is going to do be used.

1. polymorphism and inheritance?

Inheritance: one class can inherit part or all of its structure and behavior from another class.

Polymorphism (many shapes) two objects can be treated the same way but can act differently.

In short, polymorphism the program decides what method to use depending on what is its actual type.

1. early binding and late binding?

Early (or static) binding refers to compile time binding and late (or dynamic) binding refers to runtime binding (for example when you use reflection).

1. overloading and overriding?

Overloading is having the same name but different parameters.

Overriding is redefining a subclass.

2) Use either declared or actual:

a) The messages that can be sent to an object depend on its declared type.

b) The way the object responds depends on its \_\_\_actual type.

|  |  |
| --- | --- |
| public abstract class Car {  private Door \_driverDoor;  protected Engine \_engine; // a  // more properties go here  public Car() {  \_driverDoor = new Door();  // initialize other variables here  }  public abstract void move(); //b    public void startEngine() {  // code to start up engine  }  // more methods go here  } // end of class Car | public class SportsCar extends Car {  private ConvertibleTop \_top;  // more properties here  public SportsCar() {  super();  \_top = new ConvertibleTop();  // initialize other variables here  }  public void move() {  // code to move really fast  }  public void startEngine() {  // code to start engine with the  // gusto of a SportsCar! Vroom!  }  } // end of class SportsCar |
| public class Van extends Car {  public Van() {  super();  }  public void move() {  // code to move at a moderate speed  }  } // end of class Van | public class CS221Mobile extends Car {  public CS221Mobile() {  super();  }  public void move() {  // code to just pitter along!  }  } // end of class CS221Mobile |
| public class SchoolVan extends Van{  public SchoolVan () {  super();  }  public void move(){  // code to move slowly and carefully  }  }// end of class SchoolVan |  |

CS221**: Groupwork on Polymorphism** Date: \_\_\_\_\_\_\_ Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name of Others in Group: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Draw a UML of the 5 classes:

2. a) Given the declaration, **private Car chuck**, which has a declared type of Car, fill in the following:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Actual Type | Valid declaration  and initiation? | What happens with the statement: chuck.move(); |
| chuck = new Van( ); |  |  |  |
| chuck = new Car( ); |  |  |  |
| chuck = new SchoolVan ( ); |  |  |  |
| chuck = new SportsCar( ); |  |  |  |
| chuck = new CS221Mobile( ); |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| b) Fill in the following: | Variable Name | Declared Type | Actual Type | Valid? | chuck.move(); |
| private Van chuck;  chuck = new Van( ); |  |  |  |  |  |
| private Van chuck;  chuck = new SchoolVan( ); |  |  |  |  |  |
| private SportsCar chuck;  chuck = new Car( ); |  |  |  |  |  |

|  |  |
| --- | --- |
| c) If the declared type of a variable is: | The actual type can be: |
| SportCar |  |
| SchoolVan |  |
| Car |  |
| CS221Mobile |  |
| Van |  |

|  |  |
| --- | --- |
| 3. Use the following UML | Animal  Bird  Cat  FatCat |

a) Fill in the Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Variable Name | Declared Type | Actual Type | Valid? |
| private Animal Chuck;  Chuck = new Cat( ); |  |  |  |  |
| private Animal Chuck;  Chuck = new Bird( ); |  |  |  |  |
| private Animal Chuck;  Chuck = new FatCat( ); |  |  |  |  |
| private Animal Chuck;  Chuck = new Animal( ); |  |  |  |  |
| private Cat Felix;  Felix = new Animal( ); |  |  |  |  |
| private Cat Felix;  Felix = new FatCat( ); |  |  |  |  |
| private Cat Felix;  Felix = new Bird( ); |  |  |  |  |

|  |  |
| --- | --- |
| class ProgramApp{  private Animal an1, an2, an3, an4; //c  public ProgramApp {  an1 = new Cat();  an2 = new Bird();  an3 = new FatCat();  this.doCommand();  }  public void doCommand() {  this.drawpicture(an1);  this.drawpicture(an2);  this.drawpicture (an3);  }  public void drawpicture (Animal animal) { //b  animal.draw ();  // more code  }  } | What would happen if the following independent changes were made?  b)The parameter type were changed from Animal to Cat:  public void drawpicture(Cat acat){  }  c) The declared type of the instance variables were changed from Animal to Cat:  private Cat an1, an2, an3, an4; |