CS 125 - Lecture 28

Objectives: inheritance; super; dictionaries

Up next: Quiz section this week; fewer readings, more activities

1. **Discuss** with a neighbor and answer the following:

a) In object oriented programming, inheritance means:

b) What are some reasons we might use inheritance?

c) In java, inheritance is implemented using the keyword:

d) Write a public class statement for a **phone** class that subclasses a **device**

2. Remember, subclasses can**:**

a) declare a field in the subclass with the same name as the one in the superclass, thus **hiding** it

b) define a new instance method in the subclass that has the same signature as the one in the superclass, thus **overriding** it.

Are those **hidden** and overridden **members** now inaccessible?

4. instanceof operator …

```
  Object x = new Point(1,2) // defined above
  Object y = new Integer (5);
  Object z = new LabeledPoint("Hi",2,3);
//LabeledPoint extends Point to include a label.

  (x instanceof Point) -> true
  (y instanceof Point) -> false
  (z instanceof Point) -> true
  (x instanceof LabeledPoint) -> ?
```

3. class hierarchy…

```
File #1:
  public class Point {
     private double x,y;
     public double getX() { return x; }
     public double getY() { return y; }
     public String toString() { return “(“+x+”,"+y+")"; }
     // write a constructor that takes two integers: newX, newY




File #2:
   public class LabeledPoint extends Point {
      // add a new string property ‘label’


      // write a constructor that takes a string and two integers



      // override the toString method


   }
```

5. continued class hierarchy…

File #3 … Which of the following are valid?

```
  Object o = new LabeledPoint("Hi",2,3);
  Point p = (Point)o;
  LabeledPoint lp = (LabeledPoint)o;

  o.toString();
  p.toString();
  lp.toString();

  o.getX();
  p.getX();
  lp.getX();
```

6. **Part 1: Model the callerId objects …**

```
// Implements the key-value pair for a callerID system
public class CallerIdPair {
    // provide two protected instance variables:
      // one for the name and one for the phone number
    protected String name;
    protected int phoneNumber;
    // provide a constructor that takes a name and number as input
    public CallerIdPair(String newName, int newNumber) {
         this.name = newName;
         this.phoneNumber = newNumber;
    }
    // override the toString method (from where?) to provide a
    // useful string representation of the object
    public String toString() {
         String idString = "";
         idString += this.phoneNumber + ": " + this.name;
         return idString;
    }
}
```

7. **Let's build a caller id system that has the following functionality …**

```
public class CallerIdDriver {
   public static void main (String[] args) {
         //  Set up a new caller ID dictionary.
         //  A SINGLETON (only instance) of class CallerIdMap



         //  Add some friends
         callerIdDictionary.add(1236789, "Donny Trump");
         callerIdDictionary.add(4176578, "Warren Buffet”);
         callerIdDictionary.add(9873456, "Cinda Heeren");



         //  Retrieve information about my contacts
         String callerName = callerIdDictionary.getCallerName(9873456);
         System.out.println(callerName);
         System.out.println(callerIdDictionary.getCallerName(1236789));
         System.out.println(“Provide investment advice to : “ +
            callerIdDictionary.getCaller(4176578).name);
   }
}
```

8. **Part 2: Model the dictionary (MAP) of callerId objects …**

```
// Implements a MAP collection of CallerId key-value pairs
public class CallerIdMap {

      //  use an array of pairs



      //  add method: takes a key and a value and adds
      //  the pair object to the array
      public void add(int newNumber, String newName) {



      }

      //  get method:  returns a name for a given phone number



      //  get method: returns a key-value pair for a given phone number



}
```

5. **Let's build a game...**

```
public class Sprite {
    private int x,y,dir,shape;
    public void setX(int newX) { if(newX>0) this.x = newX; }
    public int getX() { return x; }
    public int getShape() {return shape;}
    public void move() { if(dir==1) x++; …}

    public void draw() {
      if(shape==1) Zen.drawImage("InkyGhost.png", x, y);
      if(shape==2) Zen.drawImage("Pacman.png", x, y);
      //    ... there has to be a better way...
   }

// Create a constructor to initialize the sprite using a string
whose format is xvalue,yvalue (i.e. comma-separated values)




}
```

6. **Write code to create sprites based on the position data in a file.**

```
public class Game {
   public static void main(String[] ) {









   }
}
```

7. **Complete** `.equals` **and write the two Ghost constructors so we can make ghosts such as :**

```
new Ghost(); // creates ghost at (1, random Y position)
new Ghost( new int[] {15,20} );  //ghost at (15,20)

public class Ghost {
  private static int count=0;
  private static int nextId() {
    count ++;  // first ghost will have an id of 1
    return count;
  }

  // each ghost has an x,y and unique id
  private int x=1 ,y=2+ (int)(Math.random()*10);
  private int id;  // Your constructor sets id to
                   // a unique value

  public String toString() {
       return "Ghost #"+id+": "+x+","+y;
  }

  public boolean equals(Object other) {
     if(other instanceOf Ghost) {
        Ghost g = (Ghost)other; // zombie
        return _______
    } else  return false;
  }
```