CS 125 - Lecture 22

Objectives: instance methods, instance variables.

Up next: MP5 out; Due in 9 days (week from Friday)

Enhanced learning under different contexts and environments

1. **Complete the boolean instance method below:**

```
Dog d1 = new Dog();
Dog d2 = new Dog();

d1.x = 10; d1.y = 20;
// d2.x = ..., d2.y = ...

class Dog {
   int x,y;

   // returns true if this dog is at same
   // x,y location as the other dog
   public boolean canSniffButt(Dog other) {

      return ____________________________

   }
}
```

2. **Create the House class so that the following code compiles and runs correctly.**

```
House h = new House();
      h.setStreet("101 Main St");

      // returns true iff houses have same
      // street string
      h.equals(otherHouse);
```

```
public class DemoClass {
   /** Class to demonstrate class variables, class methods,
   *  instance variables, instance methods  */

   // Class variables (static member variables)
   public static int classIntVar = 42;
   public __________  __________ classStringVar = "forty two";

   // Instance variables (non-static member variables)
   public int instanceIntVar = 21;
   __________ ___________ instanceStringVar = "twenty one";

   // Class method (static member subroutines/methods/functions)
   public static void classMethod() {

     //  Yes?  or No?  Why?
Y  N Sys.o.println("In classMethod, classIntVar: " + classIntVar);
Y  N Sys.o.println("In classMethod, DemoClass.classIntVar:"+DemoClass.classIntVar);

Y  N Sys.o.println("In classMethod, instanceIntVar: " + instanceIntVar);
Y  N Sys.o.println("In classMethod,DemoClass.instanceIntVar:"+DemoClass.instanceIntVar);
   }

   // Instance method (non-static member subroutines/methods/functions)
   public void instanceMethod() {
Y  N Sys.o.println("In instanceMethod, classIntVar: " + classIntVar);
Y  N Sys.o.println("In instanceMethod, DemoClass.classIntVar: " + DemoClass.classIntVar);
Y  N System.out.println("In instanceMethod, " + instanceIntVar);

Y  N System.out.println("DemoClass.instanceIntVar: " + DemoClass.instanceIntVar);
Y  N System.out.println("Inside instanceMethod, " + this.instanceIntVar);
   }

   public static void main(String[] args) {
Y  N System.out.println("In main, classIntVar: " + classIntVar);
Y  N System.out.println("In main, DemoClass.classIntVar: " + DemoClass.classIntVar);

     _______________;    // invoke the class method here

     _________._________________;   // invoke the class method here a different way

     __________ myNewObject = ______ ________;   // create an object of type DemoClass
Y  N Sys.o.println("myNewObject.instanceIntVar: " + myNewObject.instanceIntVar);
Y  N Sys.o.println("myNewObject.instanceStringVar: " +
                          myNewObject.instanceStringVar);

Y  N Sys.o.println("myNewObject.classStringVar: " + myNewObject.classStringVar);
   }
}
```

```
public class MedicalImage {
   public Picture picture;
   public Date date;
   public Location where;
.
.
.
}
```

```
public class Link {
   public int value;
   public Date date;
   public Link next;
.
.
.
}
```

```
public class Simulation {
   public Atom[] atoms;
   public double temp;
.
.
.
}
```

```
public class Atom {
   public double x;
   public double y;
   public double vx;
   public double vy;
.
.
.
}
```

a) ***Create a new Atom and set its position to (5,8); create another and set its position to (7, 10):***

```
// Inside Universe.java
public static void main(String[] args) {

}
```

b) ***How would we compare two atoms? You CANNOT edit Atom.java.***

1) *Create a class (static) method that takes two parameters 'a1, a2' of type Atom that returns true if two Atoms have exactly the same x and y.*

Hint static => no this pointer!

How would you invoke this method?

c) ***What if you CAN edit Atom.java?*** *Hint 1: think of comparing two string objects, s1.equals(s2). Hint 2: you will need to use 'this.x'*

2) *Create an instance (non-static) method 'equals' in Atom that takes a pointer to an atom and returns true if the atom has the same position as the given atom.*

```
// Atom.java continued…
```

Write some code that uses your method:

Where would your code fail if the atom parameter value was null?

d) ***Create an instance method "moving" in Atom that takes no parameters and returns true iff the atom's vx or vy values are non-zero:***

Write code to print the result from invoking the method:

e) ***Create an instance method "init" in Simulation that initializes the atoms array with 100 elements and creates 100 atoms at random locations:***

```
// Simulation.java continued…
```

How would you invoke this method:

Write code that prints the contents of the array?

```java
public class Atom {
   public double x;
   public double y;
   public double vx;
   public double vy;

   public Atom[] atoms;

   public static boolean equals(Atom a1, Atom a2) {
        return (a1.x==a2.x)&&(a1.y==a2.y);
   }

   public boolean equals(Atom other) {
            return (this.x==other.x)&&(this.y==other.y);
   }

   public boolean moving() {
            return (this.vx != 0.0)||(this.vy != 0.0);
   }

   public void init() {
      this.atoms = new Atom[100];
      for (int i=0; i<this.atoms.length; i++) {
           //  at first demo, forget this...
           //  use debugger to figure out where things go wrong - step into init()
           Atom tempAtom = new Atom();
           tempAtom.x = Math.random()*800.0;
           tempAtom.y = Math.random()*600.0;
           tempAtom.vx = Math.random()*10.0 - 5.0;  // want range of -5.0 to +5.0
           tempAtom.vy = Math.random()*10.0 - 5.0;  // want range of -5.0 to +5.0

           this.atoms[i] = tempAtom;

      }
   }

   public static void main(String[] args) {

      Atom atomObject = new Atom();  // construct a new atom object using the Atom template
      // Atom atomObject;  // construct a new atom object using the Atom template

      atomObject.x = 5.0;
      atomObject.y = 8.0;
      atomObject.vx = 0.0;
      atomObject.vy = 0.0;

      Atom otherObject = new Atom();  // construct a new atom object using the Atom template
      // Atom otherObject;  // construct a new atom object using the Atom template

      otherObject.x = 5.0;
      otherObject.y = 8.0;
      otherObject.vx = 0.0;
      otherObject.vy = 0.0;

      System.out.println(Atom.equals(atomObject, otherObject));
      System.out.println(atomObject.equals(otherObject));
      System.out.println(otherObject.x);
      System.out.println(otherObject.vx);
      System.out.println(atomObject.moving());

      atomObject.init();
      for (int i=0; i<atomObject.atoms.length; i++) {
           System.out.println("i="+i+" x = " + atomObject.atoms[i].x);
           System.out.println("i="+i+" y = " + atomObject.atoms[i].y);
           System.out.println("i="+i+" vx = " + atomObject.atoms[i].vx);
           System.out.println("i="+i+" vy = " + atomObject.atoms[i].vy);

      }
   }

   }
}
```