2. Objects and Classes :: Instantiating Objects

To declare object variables of a class we write:

```
Class object;

Point pete;
```

Once declared, to **instantiate** (create) an object of a class we use the **new** operator:

```
object = new Class([optional-args]);

pete = new Point(10, 20);  // Calls Point(double, double)
```

These two operations can be combined in one statement:

```
Class object = new Class([optional-args]);

Point patty = new Point(30, 40);  // Calls Point(double, double)
```

2. Objects and Classes :: Object Diagrams

We can represent *pete* by drawing an object diagram:

Within *pete* and *patty* the variables x and y are known as **instance variables** and the methods *Point*(), *getX*(), *getY*(), *setX*(), and *setY*() are known as **instance methods**.

2. Objects and Classes :: Accessibility

Good object oriented programming style dictates that instance variables shall always be declared as **private**.

```
public class Point {
  public double x;  // No!
  public double y;  // Never!
  ...
}
```

Instance methods may be **public**, **protected**, or **private**. A **public** instance method can be called on any object of the class from any location in the application.

```
public class Main {
  public static void main(String[] args) {
    Point pete = new Point(10, 20);
    pete.setX(50);
  }
}
```

2. Objects and Classes :: Accessibility (continued)

A **private** instance method may only be called by other methods of the same class.

```
public class C {
  private int a;  // Instance data members are always private.
  public C() {    // Constructors are always public.
    a = 0;
  }
  private void someMethod() {       // Private instance method.
    --a;
  }
  private void someOtherMethod() {  // Private instance method.
    someMethod();
  }
}

public class Main {
  public static void main(String[] args) {
    C cObject = new C();
    cObject.someMethod();
    cObject.someOtherMethod();
  }
}
```