Creating Objects

Classes and Objects

- A *class* is used to define the way that things look and behave.
  - Think of a class as a model, blueprint, template or the “factory” where your objects are created.

- An *object* is a particular instance of a class.
  - Think of objects as the things being produced by your “factory” or class

An example:

- A fraction is an expression composed of a numerator and a denominator in the form **n/d.**
- In addition, there are rules on how we perform operations (such as adding, subtracting, multiplying, dividing fractions) on such expressions. – More on this later!

Syntax - Creating our class

- In your project, create a new class called "*fraction*"
- In this class enter:

```
class fraction
{
    int num;   // instance field
    int den;
}
```

Instance fields

- In the class *fraction*, *num* and *den* are considered instance fields.
    - Instance fields are considered attributes of an object.
    - Although different objects do not necessarily need to have the same values, every object created should contain that field
        - (e.g. every fraction should have a *num* and a *den*, although the values can be different).

- The preceding class defines what a fraction should look like (consisting of a numerator and a denominator)
- fraction is a new type that can now be used in our programs

- Note: the class does not actually create any fractions. It simply shows us what a fraction should look like.

Comments

- When defining a class, all fields should be commented re: their purpose and the values that they can take on.
- If the field is a menu of options, those options should be described in full.

Syntax – Creating an object

- In the main method of our program, we can now declare an object of type *fraction*

fraction x = new fraction();

- *Declares and creates a new fraction object called x.*
- *All numerical values are automatically initialized to 0.*

Assigning values to our objects

- Once an object of type fraction is created, the numerator and denominator is initialized to 0.

- To change the values, you can use the following assignment statements.

  x.num = 2;

  x.den = 3;

An example: trace this code

```
public static void main (String[] args){
    fraction f = new fraction();
    f.num = 5;
    f.den = 6;
    fraction f2;
    f2 = new fraction();
    f2.num = 7;
    f2.den = 9;

    f2.num++;
    f2.den = f2.den+3;
    System.out.println(f2.num);
    System.out.println(f2.den);
}
```

An example: trace

```
public static void main (String[] args){
      fraction f = new fraction();
      f.num = 7;
      f.den = 9;

      fraction a;
      a = f;
      a.num = 4;
      a.den = 5;

      System.out.println(a.num);
      System.out.println(a.den);
      System.out.println(f.num);
      System.out.println(f.den);
}
```

Declaring a reference variable

- *From the previous example, we declare the following:*

  fraction y;

    - *Creates a variable y that only references fraction and does not yet refer to an object.*
        - *i.e. y is uninitialized*

  y = f;

    - *The values stored in the memory location of f is copied into the memory location of y.*
    - *Now, both y and f refer to the same object*
        - *Therefore, If we change the values in y, we change the values in f!*