Panel 1

All programs use:

- int
- double
- char
- boolean
- String

arrays of them

- 1D
- 2D
- 3D

Example: Program to work with Length such as convert from inch, cm, ...

double l = 1.0 + unit as String

One new data type

1

Panel 2

Classes

A class is the fundamental structure in Java. It is composed of two sections, fields to contain data and methods to manipulate data or perform an action. Every class represents a new reference type that can be used by other classes. Classes are defined using the syntax:

```
[public] class ClassName [extends ClassName] [implements InterfaceList]
{  /* list of fields */
   /* list of methods */;
}
```

options

where `ClassName` *is the name of the class,* `extends` *indicates that the class is derived from another class and* `implements` *indicates that the class has attributes in common with one or more interfaces.*

```
[modifier] class ClassName [extends ClassName] [implements InterfaceList]
fieldList;
returnType methodName(inputList)
    body of method
/* additional methods as necessary */
```

Representation of a Class

2

Panel 3

You can define anything as a new class:

```
public class Name
{
  // fields
  // methods
}
```

Review: Program = class + main method

Class defines new Data Type!

3

Panel 4

Software Engineering Tip: The hardest part in designing a class is not how to implement it, but to determine the fields and methods it should have. The "has-a" and "does-a" questions can help:

- If a class "has-a" property, then the class needs a field to store that information. Use field names that represent the property they are storing.
- If a class "does-a" certain action, then the class needs a method to perform that action. Use method names representative for the actions they perform.

Design a class to represent an Fraction

System.out.println(1/20 + 2/30); => 0. ??? NOT 5/6

```
class Fraction
{
  int num;
  int denom;
}
```

5 methods:
add, subtract, multiply, divide, print, reduce!

4

Panel 5

Adding fractions

$$\frac{a}{b} + \frac{c}{d} = \frac{da + bc}{bd}$$

$$\frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$$

$$\frac{num}{denom} + \frac{f.num}{f.denom}$$

5

Panel 6

Instantiation

Instantiation is the process of creating an object according to the blueprint of a class. Memory is allocated to provide storage space for the fields and methods and any necessary initialization is performed. The keyword `new` *is used to instantiate an object, using the syntax:*

```
[ClassName] referenceName = new ClassName(paramList);
```

where `ClassName` *is the name of the class providing the blueprint for the object and* `referenceName` *is the name of the object to be instantiated. Each new object gets its own set of data field.[1] Methods are created once and shared by objects instantiated from the same class.*

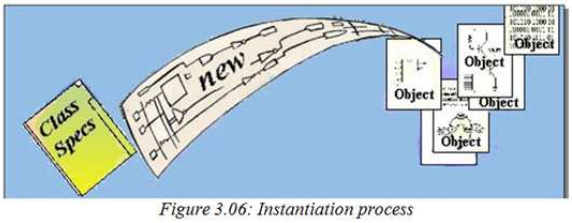

Figure 3.06: Instantiation process

6

Panel 7

Accessing Fields and Methods

Every object can access its own fields and methods by referring to them by name. Object A can access another object B if object A has a reference name pointing to object B:

- *It can use the reference name to refer to object B in its entirety.*
- *It can access the fields and methods of object B[2] using the dot operator and the syntax* `refNameForObjectB.memberOfObjectB`

Every object can refer to itself using the special keyword `this` *and to its own members using* `this.fieldOrMethodName`.

HW: add "multiply", subtract, divide to Fraction and test it!

7