UCT Department of Computer Science

Computer Science 1015F

OOP Bootcamp

Hussein Suleman
<hussein@cs.uct.ac.za>
April 2009

What is OOP?

- Object Oriented Programming
- Write programs as collections of classes:
  - Each class is a type.
  - Each class defines the behaviour of one or more objects of that class.
  - Java starts the program by invoking the main method of the “driver” class.
  - The driver's main method may then create other objects and execute other methods, and these methods may do the same, andsoforth ...

Class

- Collection of data and methods (Encapsulation).
- Template to create objects.
- Stored in a file on disk.
- Is a data type (just like int).

```
class Test {
// stuff here
}
```


Object

- Collection of data and methods.
- Instance of a class.
- Stored in memory, and only while program is running.
- Is a data value (just like 1 is an int).

```
Test anObject = new Test();
```


Instance Variables

- Variables defined in class.
- Exist as part of object as long as object exists.
- Can be accessed using dot-notation within methods of class (or outside class if public).

```
class Test {
    int mark;
    String name, subject;
}
```

Methods

- Collections of statements defined in class.
- Exist as part of object as long as object exists.
- Can be accessed using dot-notation within methods of class (or outside class if public).
- Can be overloaded.

```
class Test {
    public void printName ()
    {
        System.out.println (name);
    }
}
```


Parameters and Return Values

- Parameters send set of data values to method.
- Return values get one value back, like a function.

```
class Test {
    public void setName ( String aName )
    {
        name = aName;
    }
    public String getName ()
    {
        return name;
    }
}
```


this, toString, equals

- *this* is a reference to the current object.
  - no need to define - automatic
- *toString* is a special method that must return a String representation of the object.
  - define if needed – needed sometimes
- *equals* is a special method that must return if an object is equal to the one passed in as a parameter.
  - define if needed – needed sometimes
- Mutators (usually per instance variable) directly modify instance variables based on parameters.
  - define if needed – needed most of the time
- Accessors (usually per instance variable) return the values of instance variables.
  - define if needed – needed most of the time

Constructors

- Special method to initialise object.
- Invoked only when object is being created.
- Can be overloaded.
  - Can specify initial values as parameters.
  - Can specify no parameters and assume initial values.
  - Can specify an object to copy.

```
class Test {
    public Test ( String aName, String aSubject )
    {
        name = aName; subject = aSubject; mark=0;
    }
}
```


Information Hiding

- Declare all instance variables as **private** so they CANNOT be accessed using dot-notation except from methods in the same class.
- Declare some methods as **public** so they CAN be accessed using dot-notation from methods in other classes.

Static Variables/Methods

- Variables/Methods that can be used without an object.
- Defined in class with “static” prefix.
- Instance variables are 1-per-object, static variables are 1-per-class

```
class Test {
    private static int counter;
    public static addOne ()
    {
        counter++
    }
}
```


Automatic Boxing/Unboxing

- Convert from primitive types (int, float, etc.) to/from corresponding classes (Integer, Float, etc.)

```
Integer marksObject = new Integer (12);
Integer marksObject = 12;

int marks = marksObject;
int marks = marksObject.intValue ();

int marks = Integer.parseInt (“12”);
```


References

- Java stores objects indirectly by storing a their memory locations – primitive data is stored directly.
- Assignment causes 2 variables to reference the same object.

```
Test t1 = new Test ();
Test t2 = t1;
t1.setName (“hussein”);
System.out.println (t2.getName());
```


null

- Special value for any class type variable.
- Equivalent to zero or nothing.
- Used where variable does not reference anything particular.

Reference Parameters

- Class type objects passed as parameters are also references.
- This means a method that makes changes to its parameters will cause changes to the original objects.

```
public boolean equals ( Test another )
{
    return ((another != null) &&
            (this.name.equals (another.name)) &&
            (this.marks == another.marks) &&
            (this.subject.equals (another.subject)));
}
```

Copy Constructors

- Initialise an object to be a copy of another object, passed in as a parameter.
- Results in a deep copy, instead of a shallow copy – a completely new object, not just a reference.

```
public Test ( Test another )
{
    name = another.name;
    subject = another.subject;
    marks = another.marks;
}
```


Problem

- Write a program to manage a bank account using a BankAccount class.
- The BankAccount class should store information about a single account (name, a/c number, balance).
- The driver class must interact with the user for typical operations (withdraw, deposit, query, etc.).

Problem

- Write a program to track some basic statistics for a set of marks: minimum, maximum and average.
- The class Statistics should contain a method to add a new mark and update the instance variables appropriately.
- The driver program must submit a sequence of marks from the user to the Statistics object and output statistics at the end.

Problem

- Write a program to manage the counting of votes in an election.
- The Election class must have variables to count votes for ANC/DA/Cope/ID/etc. and associated methods.
- The Vote class must store a single vote.
- The main method must create an Election object, then create a sequence of Vote objects and send those to the Election object to be counted.

Problem

- Write a program that creates a family tree of Person objects (defined/hard-coded in the driver class).
- Each Person object should store a reference to a spouse and links to up to 3 children, with appropriate accessors and mutators.
- The tree should be printable using a single call to the toString method of the head of the family tree.

Problem

- Write a program to calculate the value of `Sin(x)` for any real value of x. Use the infinite Taylor series approximation:

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}.$$

- Extend your program to draw a `Sin(x)` graph using ASCII art.

Problem

- Write an OOP program to simulate the operation of Facebook:
  - Driver class creates an instance of Facebook and an instance of User and sends User to Facebook.
  - Facebook authenticates user and creates FacebookApp object.
  - FacebookApp object generates profile box and sends it back.
  - Facebook displays User information as well as FacebookApp profile box.