Quick summary…

So far:

- Defining classes and interfaces
- Implementing interfaces
- Using interfaces

Next:

- Access control
- Static variables / methods
- Initializing an object / Constructors

Access Control:

« Behavior is public, data is private »

Defining a class in Java

```
public class OtherFile implements IFile
{
  private char[] name;

  private char[] contents;


  public void rename() {...}

  public FileData getData(){...}

  public void setData(FileData newdata){...}

  private void compressDataStorage()
}
```

Access Control

- (none specified)
  - within the same package
- public
  - everywhere
- private:
  - only in the same class
- protected:
  - within the same package, and by subclasses

static methods

```
class File
{
    private String m_name = “”;

    private FileData m_contents = null;


    public static String addExt(String name){

        return (name + “.pdf”);

    }

  }
}
```

static methods

```
class File
{
    private String m_name = “”;

    private FileData m_contents = null;


    public static String addExt(String name){

        m_fileName = name;

        return (name + “.pdf”);

    }

}
```

Error!

static methods

```
class File
{
    private String m_name = “”;

    private static int s_count = 0;


    public static String addExt(String name){

        return (name + “.pdf”);

    }

  }
}
```

Class vs. Object

What's the difference?

Initializing an object

```
class File
{
    private String m_name = “”;

    private FileData m_contents = null;


    public File(String fileName){

        m_name = fileName;

        m_contents = null;

  }
}
```

Initializing an object

```
class File
{
    private String m_name = “”;

    private FileData m_contents = null;

    // Constructor
    public File(String fileName){

        m_name = fileName;

        m_contents = null;

    }
}
```

Initializing an object

```
class File
{
    public File(String fileName){
        m_name = fileName;
        m contents = null;
    }
    public File(){
        m_name = null;
        m_contents = null:
    }
```

Initializing an object with an array

```
class File
{
    private int[] m_pageNumbers = new int[100];


}
```

Initializing an object with an array

```
class File
{
    private int[] m_pageNumbers = null;


    public File(int NumPages){
        m_pageNumbers = new int[numPages];
    }
}
```

Creating and using objects

```
class FileSystem

{

    public static void main(String[] args){

        Folder root = new Folder();

        File homework = new File("hw-one.txt");

        root.addfile(homework);

    }

}
```

Creating and using objects

```
class FileSystem

{

    public static void main(String[] args){

        Folder root = new Folder();

        File homework = new File("hw-one.txt");

        root.addfile(homework);

    }

}
```

Creating and using objects

```
class FileSystem

{

    public static void main(String[] args){

        Folder root = new Folder();

        File homework = new File("hw-one.txt");

        root.addfile(homework);

    }

}
```

Creating and using objects

```
class FileSystem

{

    public static void main(String[] args){

        Folder root = new Folder();

        File homework = new File("hw-one.txt");

        root.addFile(homework);

    }

}
```

Java Operators

Operator	Functionality
=	assignment
+, –, *, /	plus, minus, multiplication, division
%	remainder
++, – –	increment, decrement
==, !=	test equality
<, >	less than, greater than
<=, >=	less-than-or-equal, greater-than-or-equal
<<, >>	left shift, right shift
&&, ||	logical and, logical or
~, &, ^, |	bitwise operations: complement, and, xor, or

Primitive Data Types

Name	Size	Min	Max
byte	8 bit	-128	128
short	16 bit	-32,768	32,768
int	32 bit	-2,147,483,648	2,147,483,647
long	64 bit	-9,223,372,036,854,775,808	9,223,372,036,854,775,808
float	32 bit		
double	64 bit		
boolean	1 bit	**false**	**true**
char	16 bit (unicode)	\u0000 (0)	\uffff (65535)