Today

Document Distance Problem

Java Overview

Object-oriented Programming

Programming Paradigms

Models of programming:

- Procedural (imperative) languages
- Functional languages
- Declarative languages
- Object-oriented languages

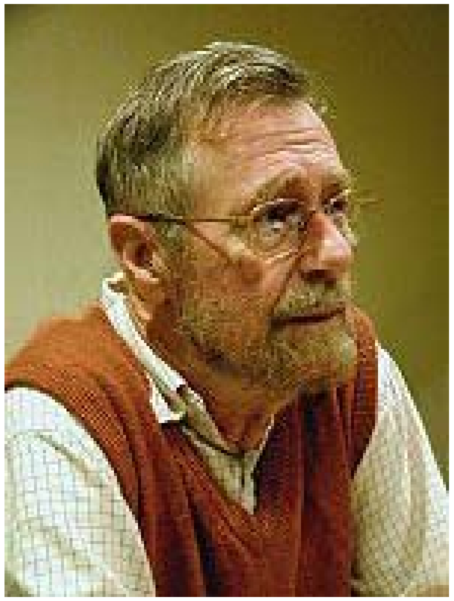
How to organize information?

How to think about a solution?

Programming Paradigms

Procedural Languages

- Examples:
  - Algol60, Fortran, COBOL, BASIC, Pascal, C
- Organization:
  - Group instructions into "procedures" or "functions"
  - Each procedure modifies the **state**.
  - Don't use GOTO statement (see…)
- Advantages:
  - Readability
  - Procedure re-use



Programming Paradigms

Functional Languages

- Examples:
  - Scheme, Lisp
- Organization:
  - Everything is a function
  - Output depends only on input
  - No state, no mutable data
- Advantages:
  - Simplicity, elegance
  - Describe what you are doing with *verbs*.
  - Focus on computation, not data manipulation

Programming Paradigms

Object-oriented Languages

- Examples:
  - Java, C++

- Advantages:
  - Near-ubiquitous in industry
  - Modular
  - Code re-use
  - Easier to iterate / develop new versions
    - Information hiding
    - Pluggable

Object-oriented Paradigm

Encapsulation

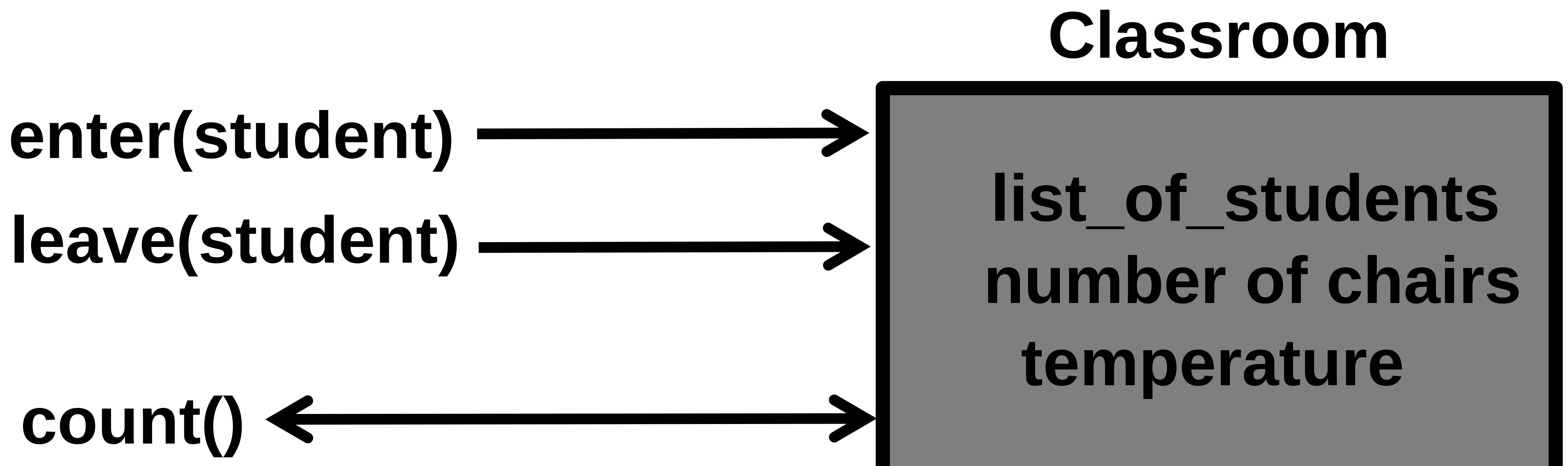
Inheritance

Polymorphism

Object-oriented Programming

Object contains:

- State (i.e., data)
- Behavior (i.e., methods for modifying the state)



How to implement a **File System**?

Files:

- Contain data
- Edited
- Rename
- Moved

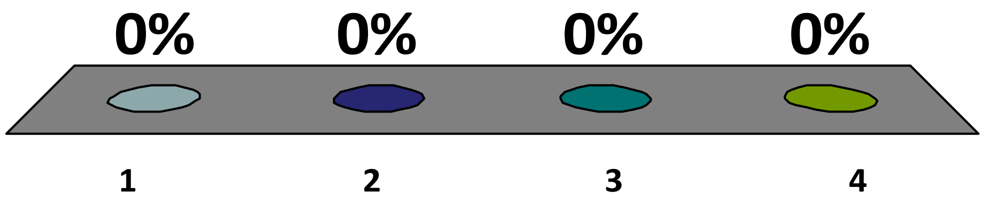
Folders:

- Contain files
- Contain folders
- Rename
- Moved

How to implement a file system?

1. file management object +
   file contents object
2. [✓ file object +
   folder object]
3. folder hierarchy +
   folder contents
4. file object

Response Counter



First principle of Java

« Everything is an object »

Defining a class in Java

```
class File
{
  String m_name = “”;
  FileData m_contents = null;


  void rename(String newName){...}
  FileData getData(){...}
  void setData(FileData newdata){...}
}
```

Objected-Oriented Java

```
class Folder
{
    String m name;
    Folder[] m_children;
    File[] m files;

    int getNumFiles(){...}
    File getFile(int i){...}

}
```

Class vs. Object

What’s the difference?

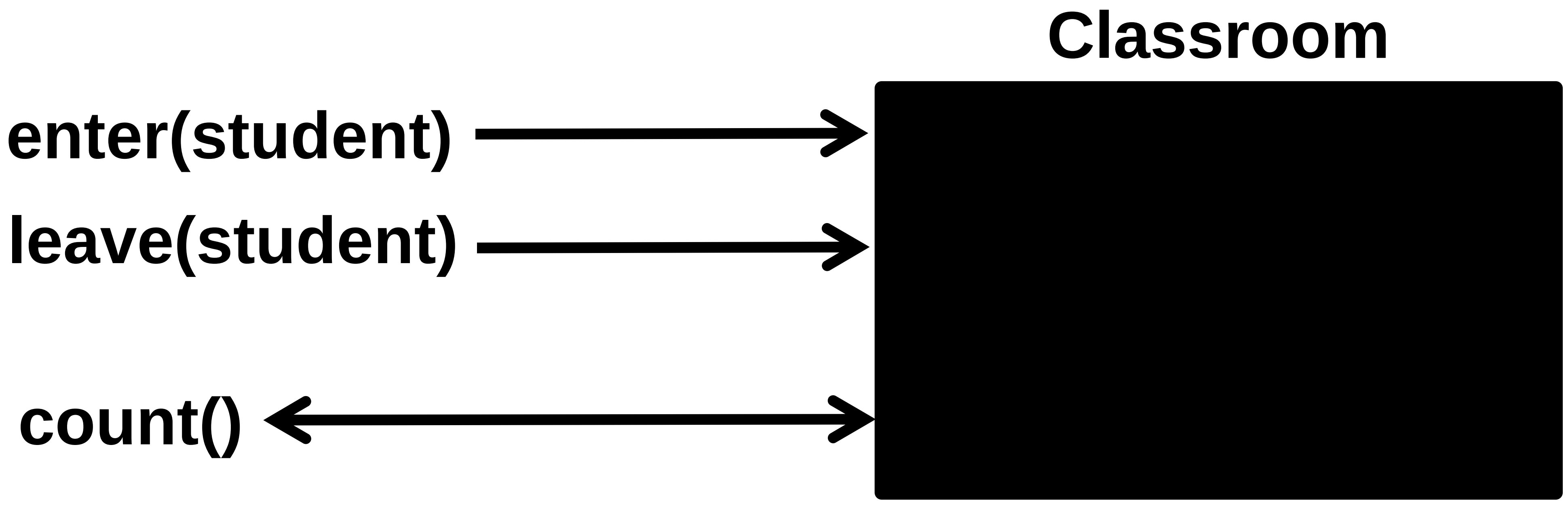
Objected-Oriented Java

```
Folder createFolder(String name) {

   Folder redFolder = new Folder(name);
   return redFolder;

}
```

Encapsulation

Interface: how you manipulate the object

Implementation: details hidden inside the object



Defining a class in Java

```
class File
{
    String m_name = “”;
    FileData m_contents = null;


    void rename(String newName){...}
    FileData getData(){...}
    void setData(FileData newdata){...}
}
```

Defining an interface

```
// Comments go here
interface IFile
{
    // Comments explain how to use interface
    void rename(String newName);

    FileData getData();

    void setData(FileData newdata);

}
```

Defining a class in Java

```
class File implements IFile
{
   String m_name = “”;
   FileData m_contents = null;


   void rename(String newName){...}
   FileData getData(){...}
   void setData(FileData newdata){...}
}
```

Defining a class in Java

```
class OtherFile implements IFile
{
  char[] name;
  char[] contents;

  FileData getData(){...}
  void setData(FileData newdata){...}
}
```

Error!

Defining a class in Java

```
class OtherFile implements IFile
{
  char[] name;
  char[] contents;


  void rename() {...}
  FileData getData(){...}
  void setData(FileData newdata){...}
}
```

Objected-Oriented Java

```
IFile copyFile(IFile oldFile) {

  File newFile = new File();

  FileData data = oldFile.getData();
  newFile.setData(data);

  return newFile;
}
```

Quick summary...

So far:

- Defining classes and interfaces
- Implementing interfaces
- Using interfaces

Next:

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