LECTURE 09: WORKING WITH FILES

greenwich.edu.vn

UNIVERSITY of GREENWICH

Alliance with FPT Education

UNIVERSITY of GREENWICH
Alliance with FPT Education

PLAN

- File & Stream
- Binary Files
- Text Files
- Serialization

UNIVERSITY of GREENWICH
Alliance with FPT Education

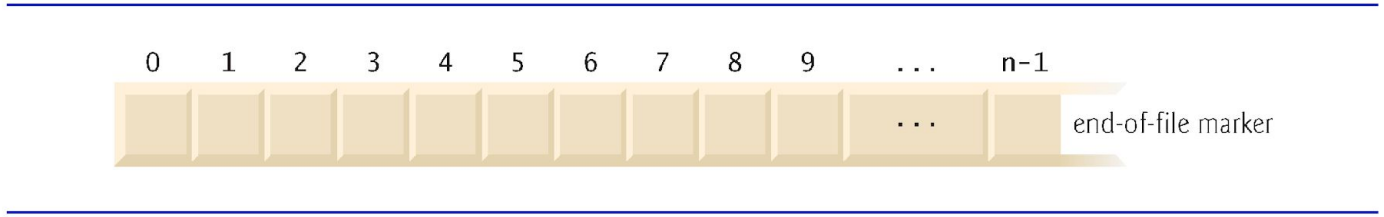
Introduction

- Data stored in variables and arrays is temporary
  - It's lost when a local variable goes out of scope or when the program terminates
- For long-term retention of data, computers use files.
- Computers store files on secondary storage devices
  - hard disks, optical disks, flash drives and magnetic tapes.
- Data maintained in files is persistent data because it exists beyond the duration of program execution.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Files and Streams

- Java views each file as a sequential stream of bytes
- Every operating system provides a mechanism to determine the end of a file, such as an end-of-file marker or a count of the total bytes in the file that is recorded in a system-maintained administrative data structure.
- A Java program simply receives an indication from the operating system when it reaches the end of the stream



UNIVERSITY of GREENWICH
Alliance with FPT Education

Files and Streams (cont.)

- File streams can be used to input and output data as bytes or characters.
- Streams that input and output bytes are known as byte-based streams, representing data in its binary format.
- Streams that input and output characters are known as character-based streams, representing data as a sequence of characters.
- Files that are created using byte-based streams are referred to as binary files.
- Files created using character-based streams are referred to as text files. Text files can be read by text editors.
- Binary files are read by programs that understand the specific content of the file and the ordering of that content.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Files and Streams (cont.)

- A Java program opens a file by creating an object and associating a stream of bytes or characters with it.
  - Can also associate streams with different devices.
- Java creates three stream objects when a program begins executing
  - System.in (the standard input stream object) normally inputs bytes from the keyboard
  - System.out (the standard output stream object) normally outputs character data to the screen
  - System.err (the standard error stream object) normally outputs character-based error messages to the screen.
- Class System provides methods setIn, setOut and setErr to redirect the standard input, output and error streams, respectively.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Files and Streams (cont.)

- Java programs perform file processing by using classes from package java.io.
- Includes definitions for stream classes
  - FileInputStream (for byte-based input from a file)
  - FileOutputStream (for byte-based output to a file)
  - FileReader (for character-based input from a file)
  - FileWriter (for character-based output to a file)
- You open a file by creating an object of one these stream classes. The object's constructor opens the file.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Files and Streams (cont.)

- Can perform input and output of objects or variables of primitive data types without having to worry about the details of converting such values to byte format.
- To perform such input and output, objects of classes ObjectInputStream and ObjectOutputStream can be used together with the byte-based file stream classes FileInputStream and FileOutputStream.
- The complete hierarchy of classes in package java.io can be viewed in the online documentation at
- http://download.oracle.com/javase/6/docs/api/java/io/package-tree.html

UNIVERSITY of GREENWICH
Alliance with FPT Education

Files and Streams (cont.)

- Class File provides information about files and directories.
- Character-based input and output can be performed with classes Scanner and Formatter.
  - Class Scanner is used extensively to input data from the keyboard. This class can also read data from a file.
  - Class Formatter enables formatted data to be output to any text-based stream in a manner similar to method System.out.printf.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Class File

- Class File provides four constructors.
- The one with a String argument specifies the name of a file or directory to associate with the File object.
  - The name can contain path information as well as a file or directory name.
  - A file or directory's path specifies its location on disk.
  - An absolute path contains all the directories, starting with the root directory, that lead to a specific file or directory.
  - A relative path normally starts from the directory in which the application began executing and is therefore "relative" to the current directory.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Class File (cont.)

- The constructor with two String arguments specifies an absolute or relative path and the file or directory to associate with the File object.
- The constructor with File and String arguments uses an existing File object that specifies the parent directory of the file or directory specified by the String argument.
- The fourth constructor uses a URI object to locate the file.
  - A Uniform Resource Identifier (URI) is a more general form of the Uniform Resource Locators (URLs) that are used to locate websites.
- Figure 17.2 lists some common File methods. The
- http://download.oracle.com/javase/6/docs/api/java/io/File.html

UNIVERSITY of GREENWICH
Alliance with FPT Education

Low-Level File I/O

- To read data from or write data to a file, we must create one of the Java stream objects and attach it to the file.
- A stream is a sequence of data items, usually 8-bit bytes.
- Java has two types of streams: an input stream and an output stream.
- An input stream has a source form which the data items come, and an output stream has a destination to which the data items are going.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Streams for Low-Level File I/O

- FileOutputStream and FileInputStream are two stream objects that facilitate file access.
- FileOutputStream allows us to output a sequence of bytes; values of data type byte.
- FileInputStream allows us to read in an array of bytes.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Sample: Low-Level File Output

```
//set up file and stream
File  outFile   = new File("sample1.data");

FileOutputStream
      outStream = new FileOutputStream( outFile );

//data to save
byte[] byteArray = {10, 20, 30, 40,
                    50, 60, 70, 80};

//write data to the stream
outStream.write( byteArray );

//output done, so close the stream
outStream.close();
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Sample: Low-Level File Input

```
//set up file and stream
File         inFile   = new File("sample1.data");
FileInputStream inStream = new FileInputStream(inFile);

//set up an array to read data in
int     fileSize  = (int)inFile.length();
byte[] byteArray = new byte[fileSize];

//read data in and display them
inStream.read(byteArray);
for (int i = 0; i < fileSize; i++) {
    System.out.println(byteArray[i]);
}

//input done, so close the stream
inStream.close();
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

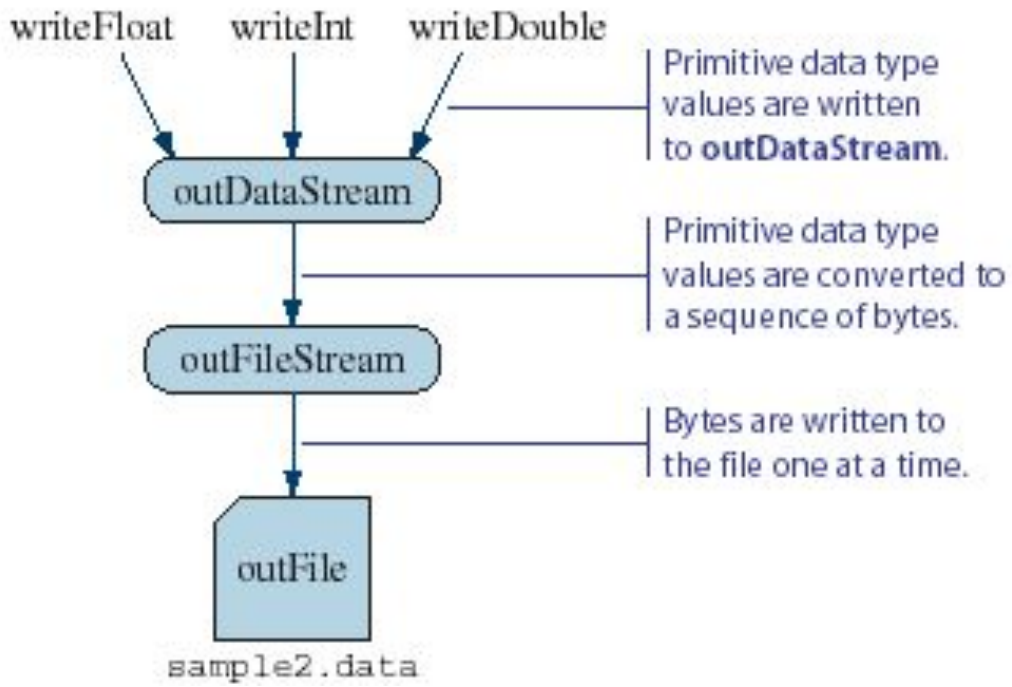
Streams for High-Level File I/O

- FileOutputStream and DataOutputStream are used to output primitive data values
- FileInputStream and DataInputStream are used to input primitive data values
- To read the data back correctly, we must know the order of the data stored and their data types

Setting up DataOutputStream

- A star

```
File             outFile       = new File( "sample2.data" );
FileOutputStream outFileStream = new FileOutputStream(outFile);
DataOutputStream outDataStream = new DataOutputSteam(outFileStream);
```



UNIVERSITY of GREENWICH
Alliance with FPT Education

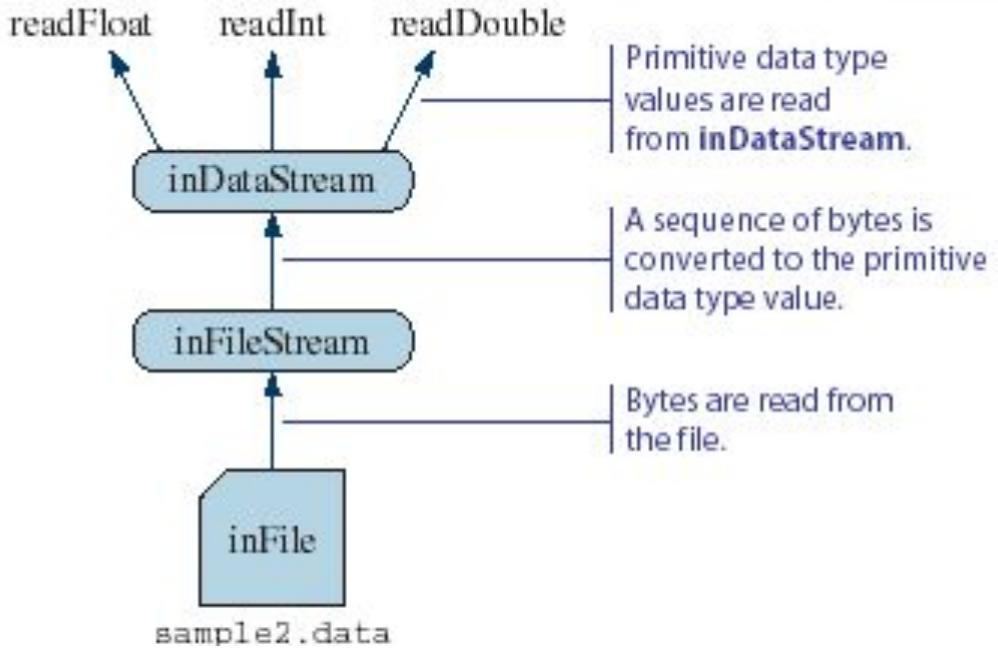
Sample Output

```
import java.io.*;
class Ch12TestDataOutputStream {
    public static void main (String[] args) throws IOException {

    . . . //set up outDataStream

    //write values of primitive data types to the stream
    outDataStream.writeInt(987654321);
    outDataStream.writeLong(11111111L);
    outDataStream.writeFloat(22222222F);
    outDataStream.writeDouble(3333333D);
    outDataStream.writeChar('A');
    outDataStream.writeBoolean(true);

    //output done, so close the stream
    outDataStream.close();
    }
}
```

Setting up DataInputStream

- A star

```
File            inFile       = new File( "sample2.data" );
FileInputStream inFileStream = new FileInputStream(inFile);
DataInputStream inDataStream = new DataInputSteam(inFileStream);
```



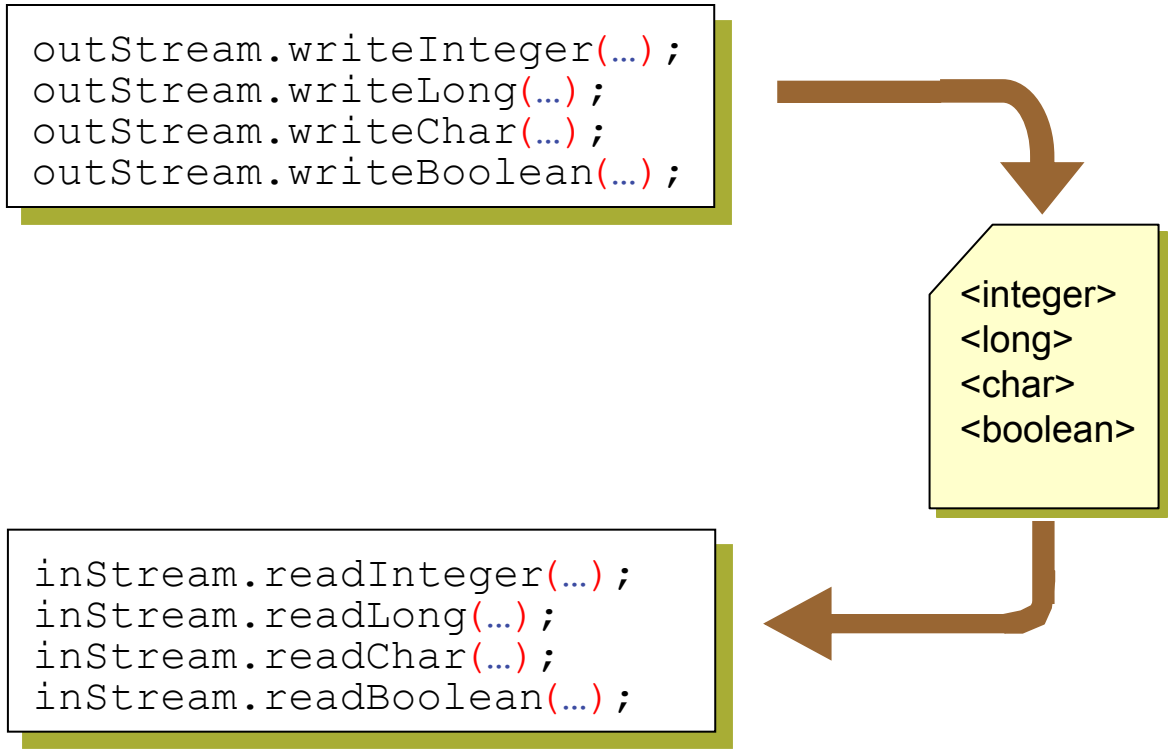
UNIVERSITY of GREENWICH
Alliance with FPT Education

Sample Input

```
import java.io.*;
class Ch12TestDataInputStream {
    public static void main (String[] args) throws IOException {

        . . . //set up inDataStream

        //read values back from the stream and display them
        System.out.println(inDataStream.readInt());
        System.out.println(inDataStream.readLong());
        System.out.println(inDataStream.readFloat());
        System.out.println(inDataStream.readDouble());
        System.out.println(inDataStream.readChar());
        System.out.println(inDataStream.readBoolean());

        //input done, so close the stream
        inDataStream.close();
    }
}
```

Reading Data Back in Right Order

- The order of write and read operations must match in order to read the stored primitive data back correctly.



UNIVERSITY of GREENWICH
Alliance with FPT Education

Text File Input and Output

- Instead of storing primitive data values as binary data in a file, we can convert and store them as a string data.
  - This allows us to view the file content using any text editor
- To output data as a string to file, we use a PrintWriter object
- To input data from a textfile, we use FileReader and BufferedReader classes
  - From Java 5.0 (SDK 1.5), we can also use the Scanner class for inputting textfiles

UNIVERSITY of GREENWICH
Alliance with FPT Education

Sample Text File Output

```
import java.io.*;
class Ch12TestPrintWriter {
    public static void main (String[] args) throws IOException {

    //set up file and stream
    File outFile = new File("sample3.data");
    FileOutputStream outFileStream
            = new FileOutputStream(outFile);
    PrintWriter outStream = new PrintWriter(outFileStream);

    //write values of primitive data types to the stream
    outStream.println(987654321);
    outStream.println("Hello, world.");
    outStream.println(true);

    //output done, so close the stream
    outStream.close();
   }
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Sample Text File Input

```
import java.io.*;
class Ch12TestBufferedReader {

    public static void main (String[] args) throws IOException {

    //set up file and stream
    File inFile = new File("sample3.data");
    FileReader fileReader = new FileReader(inFile);
    BufferedReader bufReader = new BufferedReader(fileReader);
    String str;

    str = bufReader.readLine();
    int i = Integer.parseInt(str);

    //similar process for other data types

    bufReader.close();
     }
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Sample Text File Input with Scanner

```
import java.io.*;

class Ch12TestScanner {

    public static void main (String[] args) throws IOException {

    //open the Scanner
    Scanner scanner = new Scanner(new File("sample3.data"));

    //get integer
    int i = scanner.nextInt();

    //similar process for other data types

    scanner.close();
     }
}
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Object Serialization

- To read an entire object from or write an entire object to a file, Java provides object serialization.
- A serialized object is represented as a sequence of bytes that includes the object's data and its type information.
- After a serialized object has been written into a file, it can be read from the file and deserialized to recreate the object in memory.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Object Serialization (cont.)

- Classes ObjectInputStream and ObjectOutputStream, which respectively implement the ObjectInput and ObjectOutput interfaces, enable entire objects to be read from or written to a stream.
- To use serialization with files, initialize ObjectInputStream and ObjectOutputStream objects with FileInputStream and FileOutputStream objects.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Object Serialization (cont.)

- ObjectOutput interface method writeObject takes an Object as an argument and writes its information to an OutputStream.
- A class that implements ObjectOuput (such as ObjectOutputStream) declares this method and ensures that the object being output implements Serializable.
- ObjectInput interface method readObject reads and returns a reference to an Object from an InputStream.
  - After an object has been read, its reference can be cast to the object's actual type.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Object Serialization (cont.)

- Objects of classes that implement interface Serializable can be serialized and deserialized with ObjectOutputStreams and ObjectInputStreams.
- Interface Serializable is a tagging interface.
  - It does not contain methods.
- A class that implements Serializable is tagged as being a Serializable object.
- An ObjectOutputStream will not output an object unless it is a Serializable object.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Saving Objects

```
File                    outFile
                            = new File("objects.data");
FileOutputStream    outFileStream
                            = new FileOutputStream(outFile);
ObjectOutputStream outObjectStream
                            = new
ObjectOutputStream(outFileStream);
```

```
Person person = new Person("Mr. Espresso", 20, 'M');

outObjectStream.writeObject( person );
```

```
account1        = new Account();
bank1             = new Bank();

outObjectStream.writeObject( account1 );
outObjectStream.writeObject( bank1    );
```

Could save objects from the different classes.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Reading Objects

```
File                        inFile
                                = new File("objects.data");
FileInputStream     inFileStream
                                = new FileInputStream(inFile);
ObjectInputStream inObjectStream
                                = new
ObjectInputStream(inFileStream);
```

```
Person person
      = (Person) inObjectStream.readObject( );
```

Must type cast to the correct object type.

```
Account account1
       = (Account) inObjectStream.readObject( );
Bank    bank1
         = (Bank) inObjectStream.readObject( );
```

Must read in the correct order.

UNIVERSITY of GREENWICH
Alliance with FPT Education

Saving and Loading Arrays

- Instead of processing array elements individually, it is possible to save and load the whole array at once.

```
Person[] people = new Person[ N ];
                    //assume N already has a value

//build the people array
. . .
//save the array
outObjectStream.writeObject ( people );
```

```
//read the array

Person[ ] people = (Person[]) inObjectStream.readObject( );
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

Model View Controller

- Model designs based on MVC architecture follow the MVC design pattern and they separate the application logic from the user interface when designing software.
  - Model — Responsible for managing the data of the application. It responds to the request from the view and it also responds to instructions from the controller to update itself.
  - View — Defines the presentation of the application
  - Controller — Manages the flow of the application

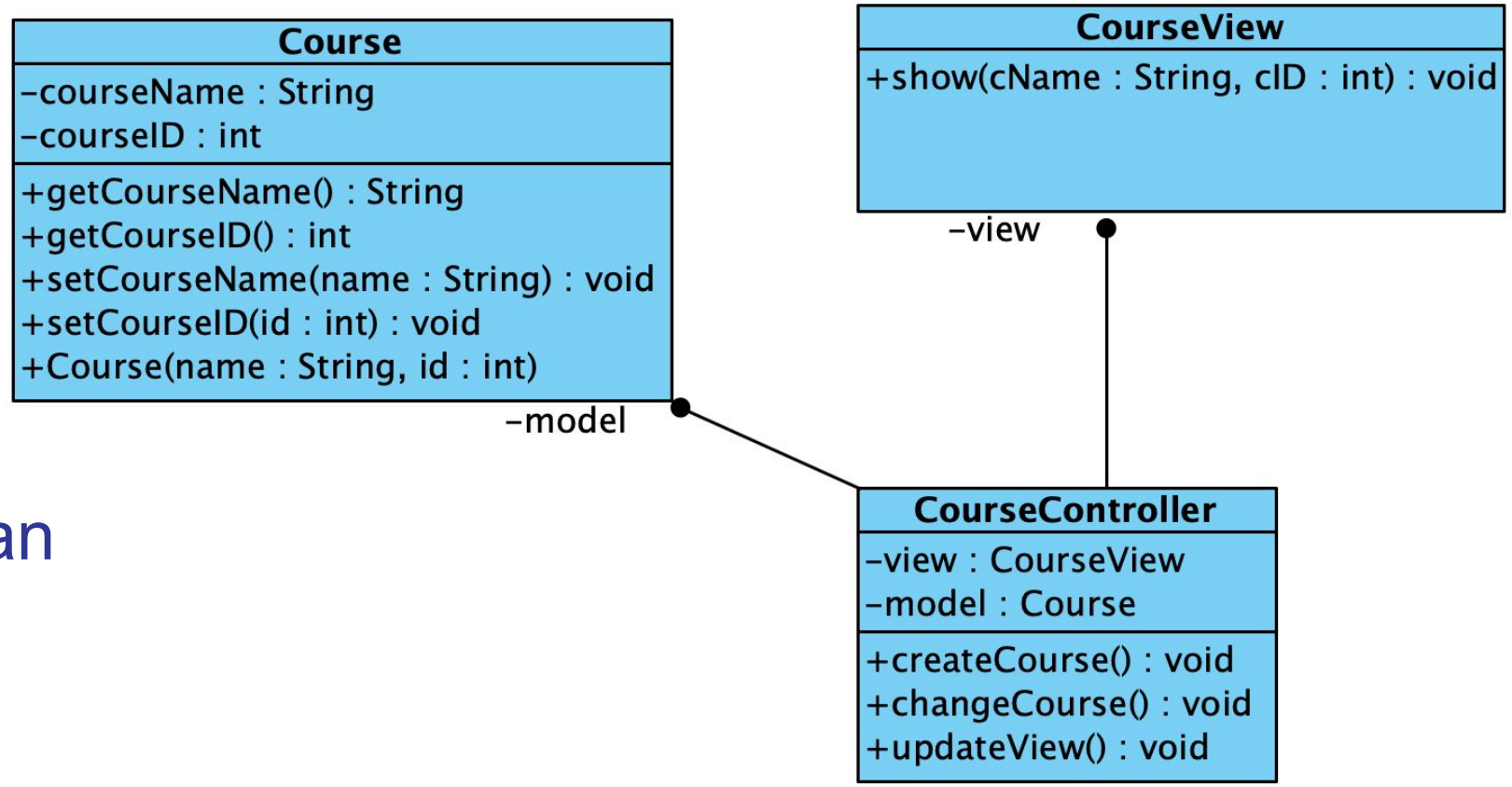
UNIVERSITY of GREENWICH
Alliance with FPT Education

Model View Controller

- MVC architecture offers a lot of advantages for a programmer when developing applications, which include:
  - Multiple developers can work with the three layers (Model, View, and Controller) simultaneously
  - Offers improved scalability, that supplements the ability of the application to grow
  - As components have a low dependency on each other, they are easy to maintain
  - A model can be reused by multiple views which provides reusability of code
  - Adoption of MVC makes an application more expressive and easy to understand
  - Extending and testing of the application becomes easy

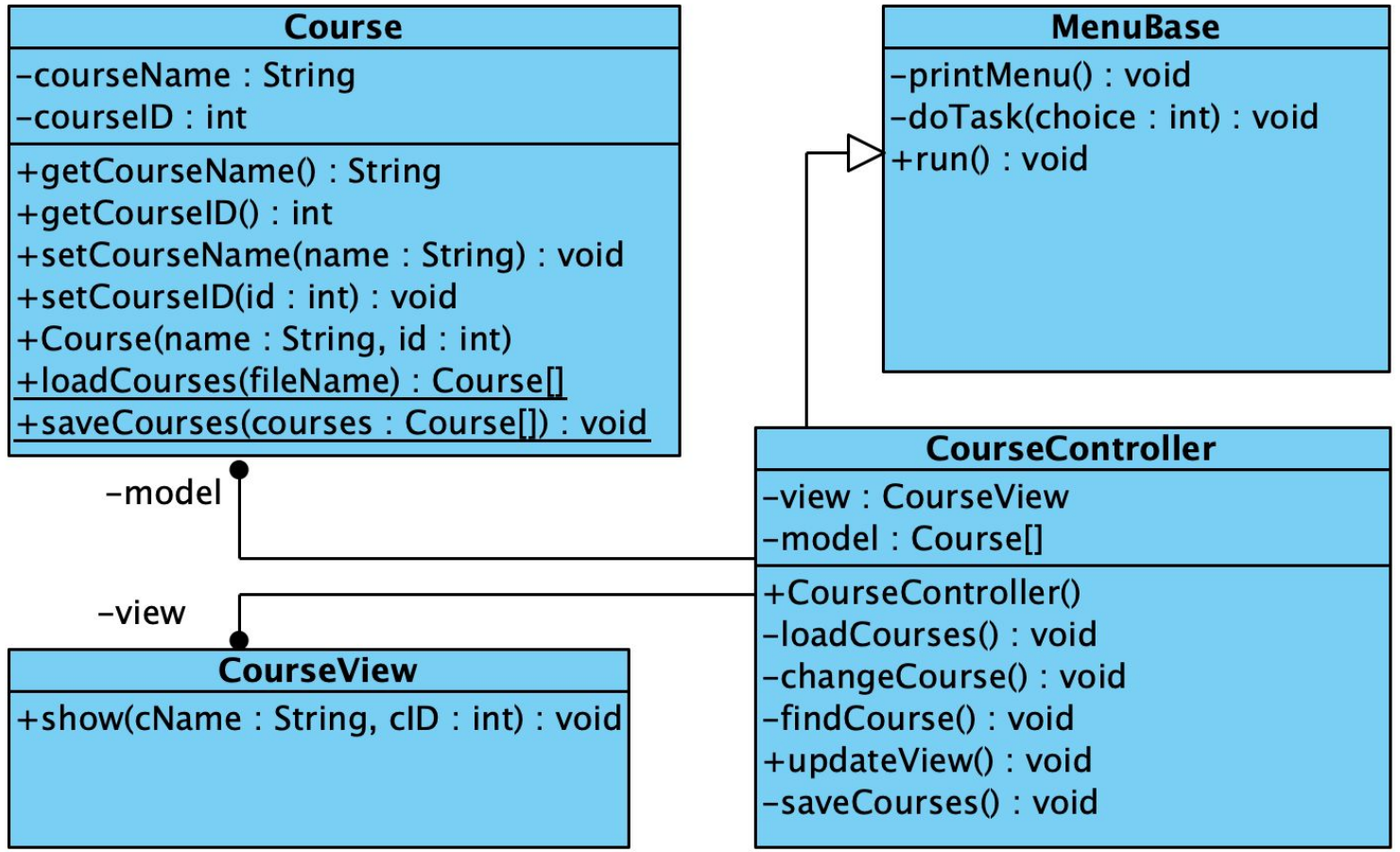
Simple Model View Controller Example

- Class Course is a simple model
- Class Controller is responsible for businesses: create a course, change a course
- Class CourseView can show a course to console



Model View Controller vs File

- Class Course
  - represents a model (course) and has responsible of working with data layer (file)
- Class CourseController inherits from MenuBase which runs by printing menu and do tasks based on user choice



Model View Controller vs File

- Class CourseController inherits from MenuBase which runs by printing menu and do tasks based on user choice
  - load/save courses from/to file
  - find a course
  - change a course