Comp 401
Abstract Classes

Instructor: Prasun Dewan

Prerequisite

- Inheritance

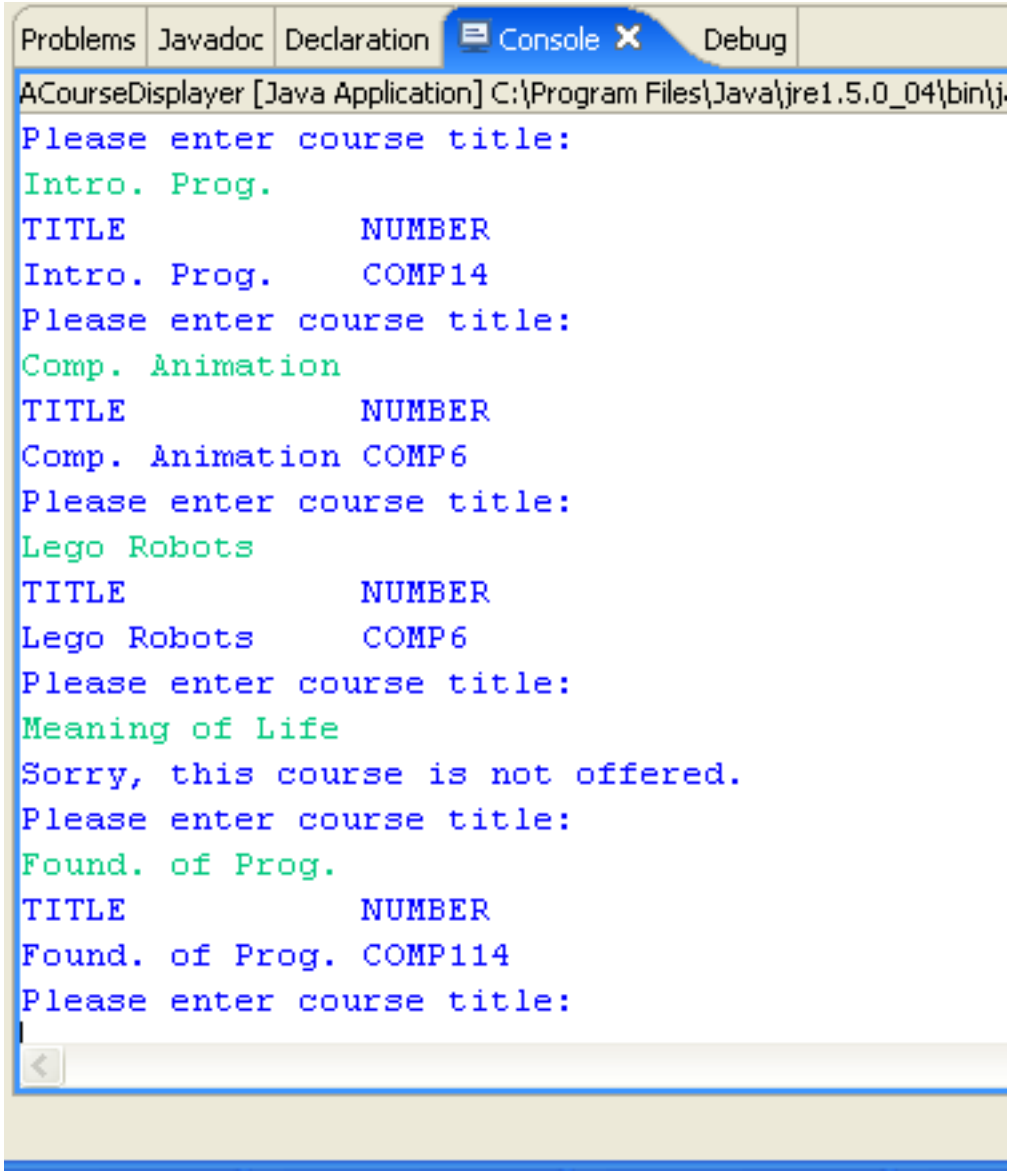
Topics

- Top-Down vs. Bottom-Up Inheritance
- Abstract Classes

Course Displayer

- User inputs course title
- Program displays course number, if offered.
- Based on old UNC numbering scheme
  - Example does not work with the new one!

Course Displayer User Interface



Course Displayer Classes

- Course List
  - Stored Courses and Matches title
- Main class
  - Creates course list and adds course instances
  - Prints courses
    - Combines dept and number into one string
- Course class(es)?

Alternative 1: One Course Class

- Course
  - Properties
    - Title, Dept, Number
  - Variables:
    - Title, Dept, Number
- Not appropriate for freshman seminar
  - Number is not a variable!

Alternative 2: Two Course Classes

- Regular Course
  - Properties
    - Title, Dept, Number
  - Variables:
    - Title, Dept, Number
- Freshman Seminar
  - Properties
    - Title, Dept, Number
  - Variables:
    - Title, Dept
  - Constant
    - Number
- But some variables/properties are common
  - Title, Dept

Alternative 3: Three Course Classes

- Additional class for common variables
  - Title, Dept
- Two course classes inherit from it
  - Regular course adds number variable and property
  - Freshman seminar adds number constant and property

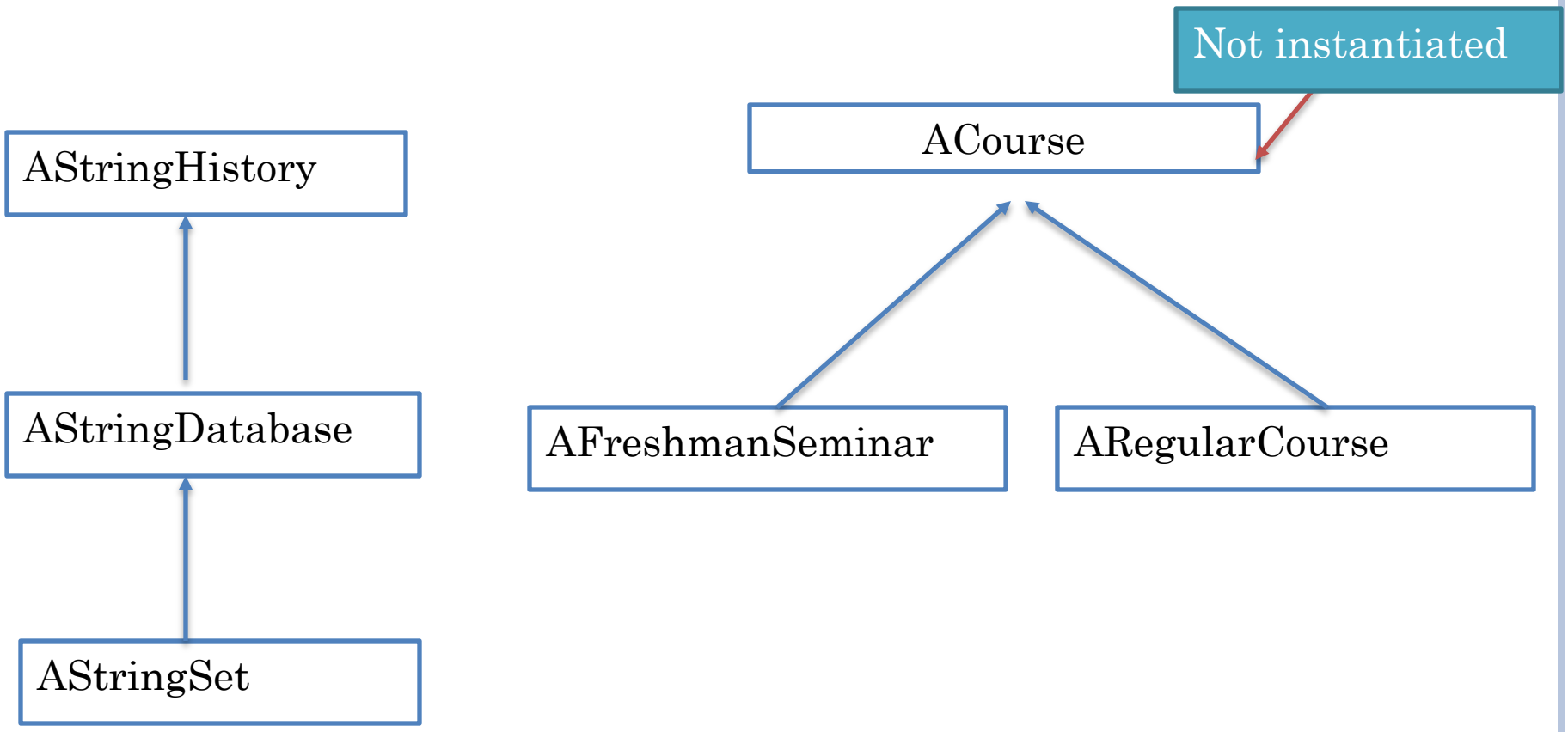
ACourse

- Should not be instantiated
- Cannot have course without a number!
- Always some subclass is instantiated
- Declared as abstract class

Abstract classes

- Java ensures they are not instantiated
- Abstract vs. regular class ⇔ Car vs. Honda Accord, Mammal vs. Homo Sapiens,

Bottom-up vs. Top-down Inheritance Hierarchy



ACourse

```
package courses;
public abstract class ACourse  {
        String title, dept;
        public ACourse (String theTitle, String theDept) {
                title = theTitle;
                dept = theDept;
        }
        public String getTitle() {
                return title;
        }
        public String getDepartment() {
                return dept;
        }
}
```

ARegularCourse

```
public class ARegularCourse extends ACourse implements Course {
        int courseNum;
        public ARegularCourse (String theTitle, String theDept, int
theCourseNum) {
                super (theTitle, theDept);
                courseNum = theCourseNum;
        }
        public int getNumber() {
                return courseNum;
        }
}
```

AFreshmanSeminar

```
public class AFreshmanSeminar extends ACourse implements
FreshmanSeminar {
        public AFreshmanSeminar (String theTitle, String theDept) {
                super (theTitle, theDept);
        }
        public int getNumber() {
                return SEMINAR_NUMBER;
        }
}
```

Course Interface

```
public interface Course {
        public String getTitle();
        public String getDepartment();
        public int getNumber();
}
```

Freshman Seminar Interface

```
public interface FreshmanSeminar extends Course {
        public final int SEMINAR_NUMBER = 6;
}
```

Alternative Abstract Class

```
package courses;
public abstract class ACourse implements Course{
        String title, dept;
        public ACourse (String theTitle, String theDept) {
                title = theTitle;
                dept = theDept;
        }
        public String getTitle() {
                return title;
        }
        public String getDepartment() {
                return dept;
        }
}
```

```
public interface Course {
        public String getTitle();
        public String getDepartment();
        public int getNumber();
}
```

Abstract Class Implementing Interface

```
package courses;
public abstract class ACourse implements Course{
        String title, dept;
        public ACourse (String theTitle, String theDept) {
                title = theTitle;
                dept = theDept;
        }
        public String getTitle() {
                return title;
        }
        public String getDepartment() {
                return dept;
        }
}
```

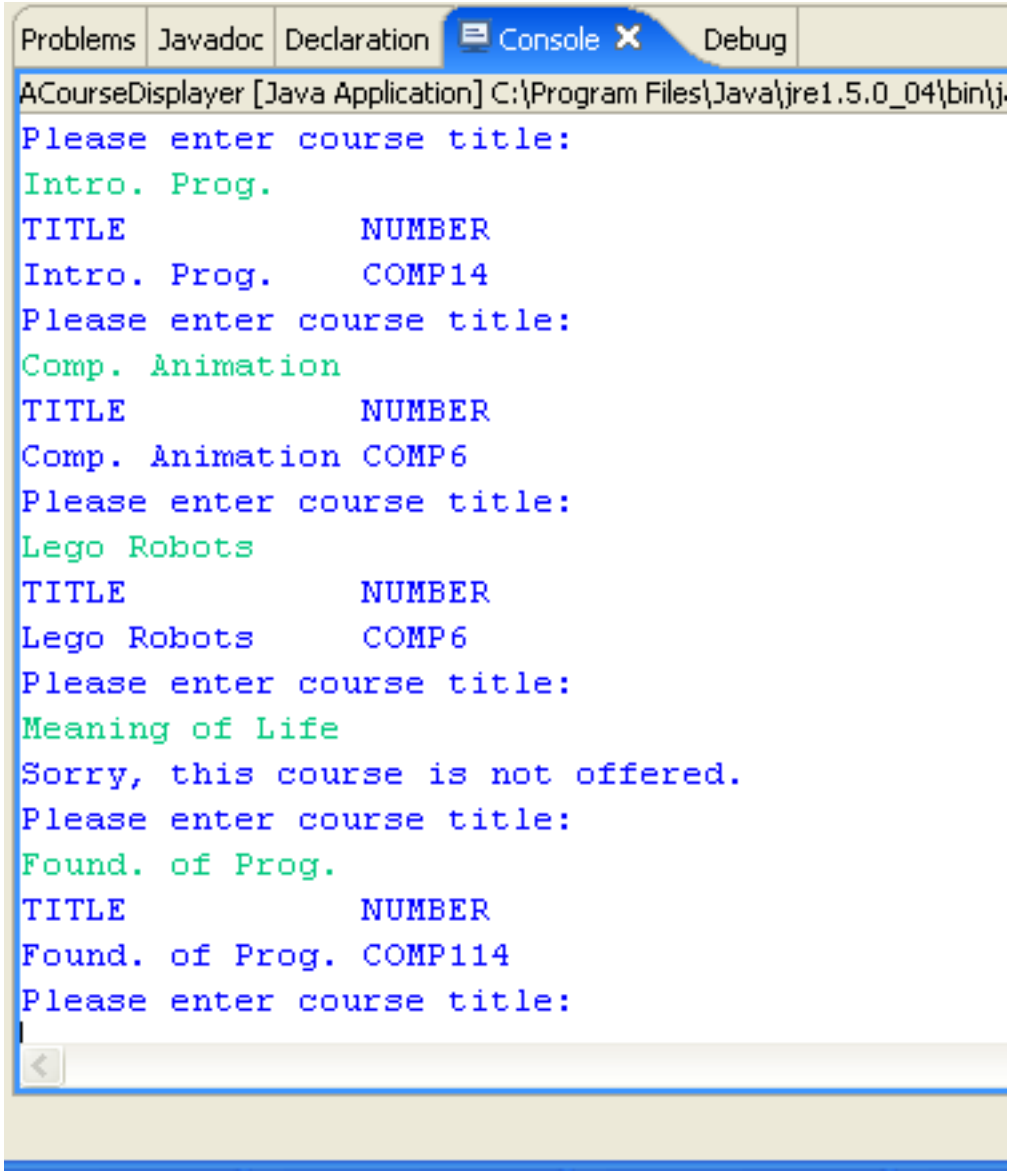
```
public interface Course {
        public String getTitle();
        public String getDepartment();
        public int getNumber();
}
```

Alternative ARegularCourse

```
public class ARegularCourse extends ACourse {
        int courseNum;
        public ARegularCourse (String theTitle, String theDept, int
theCourseNum) {
                super (theTitle, theDept);
                courseNum = theCourseNum;
        }
        public int getNumber() {
                return courseNum;
        }
}
```

A class inherits all the interfaces of its superclass(es)

Course List?


Problems
Javadoc
Declaration
Console
Debug
ACourseDisplayer [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\j
Please enter course title:
Intro. Prog.
TITLE NUMBER
Intro. Prog. COMP14
Please enter course title:
Comp. Animation
TITLE NUMBER
Comp. Animation COMP6
Please enter course title:
Lego Robots
TITLE NUMBER
Lego Robots COMP6
Please enter course title:
Meaning of Life
Sorry, this course is not offered.
Please enter course title:
Found. of Prog.
TITLE NUMBER
Found. of Prog. COMP114
Please enter course title:

CourseList

```
public interface CourseList  {
        public void addElement(Course element);
        public Course matchTitle (String theTitle);
}
```

ACourseList: Storing Elements

```
public class ACourseList implements CourseList {
        final int MAX_SIZE = 50;
        Course[] contents = new Course[MAX_SIZE];
        int size = 0;
        int size() {return size; }
        boolean isFull() {return size == MAX_SIZE;  }
        public void addElement(Course element) {
                if (isFull())
                        System.out.println("Adding item to a full
collection");
                else {
                        contents[size] = element;
                        size++;
                }
        }
```

ACourseList: Search Function

```
public Course matchTitle (String theTitle) {
        for (int courseIndex = 0; courseIndex < size; courseIndex++) {
                if (contents[courseIndex].getTitle().equals(theTitle))
                        return contents[courseIndex];
                }
        }
        return null;
}
```

Main class: Main method

```
public class ACourseDisplayer    {
    public static void main(String[] args) {
            fillCourses();
            while (true) {
                System.out.println("Please enter course title:");
                String inputLine = System.console().readLine();
                if (inputLine.equals("."))
                        break;
                Course matchedCourse = courses.matchTitle(inputLine);
                if (matchedCourse == null)
                        System.out.println("Sorry, this course is not offered.");
                else {
                        printHeader();
                        print (matchedCourse);
                }
        }
}
```

Main class: Filling List

```
static CourseList courses = new ACourseList();
static void fillCourses() {
   courses.addElement(new ARegularCourse ("Intro. Prog.",  "COMP", 14));
   courses.addElement(new ARegularCourse ("Found. of Prog.", "COMP", 114));
   courses.addElement(new AFreshmanSeminar("Comp. Animation", "COMP"));
   courses.addElement(new AFreshmanSeminar("Lego Robots",  "COMP"));
}
```

Main class: printing

```
static void printHeader() {
        System.out.println ("TITLE          "  + "NUMBER    ");
}

static void print (Course course) {
        System.out.println(
                                        course.getTitle() + "          " +
                                        course.getDepartment() +
                                        course.getNumber()
                                        );

}
```