COMPSCI 230 Tutorial 12 Review

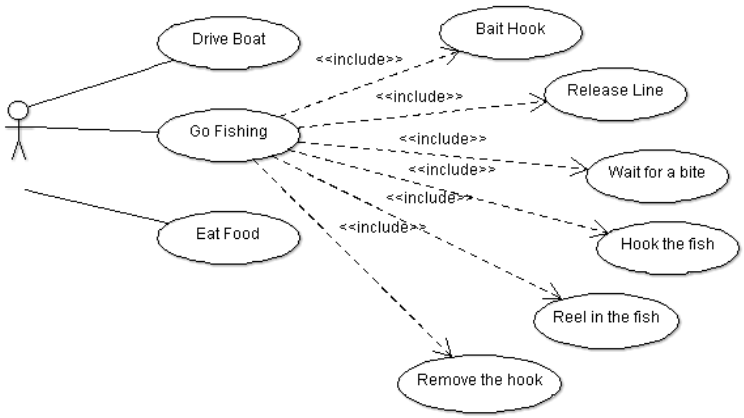
Theme A: the object-oriented programming paradigm

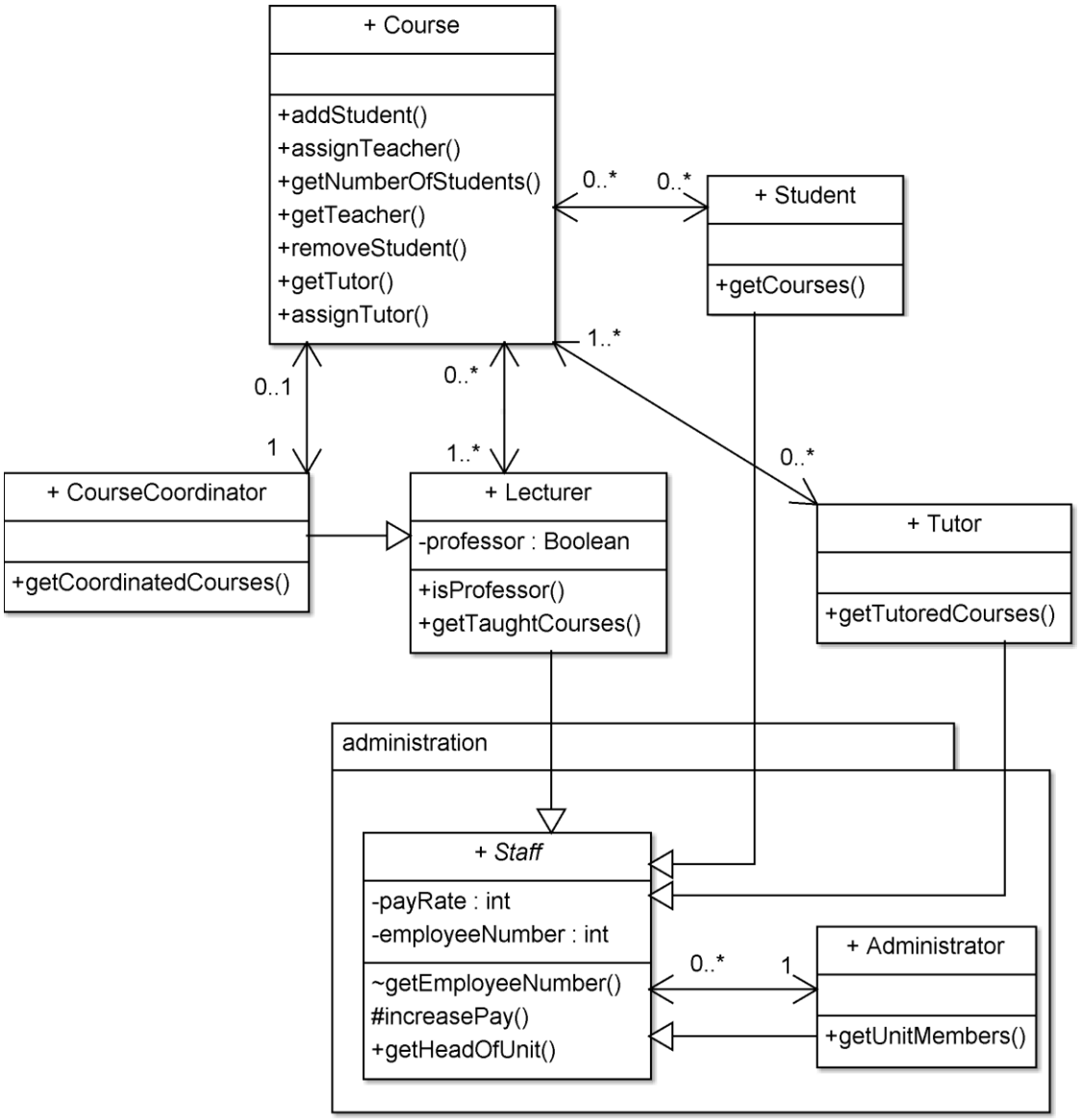
1) Remind ourselves of what the following keywords for OO in java mean:

public	Can be accessed outside the class (anywhere).
private	Can only be accessed inside the class it is declared in.
protected	Can be accessed by child classes and classes in the same package.
static	Is shared by all instances of the declaring class.
(No Modifier)	Can be accessed by classes in the same package.

2) Bob is a fisherman, he has his own boat! When he is fishing, he does three things: drive the boat, go fishing, and eat his lunch.

Draw a use-case diagram to represent Bob's fishing. Use «include» to provide a little more detail for the go-fishing part. Note: a use-case diagram should have only a few cases; the one below indicates some of the many possibilities for the inclusion.





3) Refer to the above class diagram. List the instance variables of the Lecturer Class.

```
professor: Boolean
employeeNumber: int
payRate: int
myCourses: Collection<Courses>
myAdministrator: Administrator
```

4) Describe the relationships between Course, Lecturer, and CourseCoordinator.

CourseCoordinator 'is-a' Lecturer. It is a subclass of Lecturer and thus has all the attributes of Lecturer.

A course must have at least one Lecturer. There are no restrictions on how many classes a lecturer can teach.

A course must have exactly 1 CourseCoordinator. There are no restrictions on how many courses are coordinated by a CourseCoordinator.

Hint: See Appendix for Class Diagram summary sheet.

Theme B: frameworks

5) Write an event listener so that the Tutor gets a $1.00 pay rise every time the button is clicked. You should "program by example": you'll find a suitable example in the appendix.

```
Tutor tut = new Tutor();
JButton b = new JButton("Moar Money!");
```

5) What does separation of model and view mean? Name one advantage of this.

Separating the model and view means that the GUI elements (the View) are developed and maintained independently of from the data that will be displayed (the Model).

Some advantages: the model can be displayed easily and consistently in multiple windows, and in multiple representations (e.g. a pie graph, bar graph or table)

Theme C: software quality

8) Write 3 test cases for the increasePay() method in the Administrator class. You should assume that the payRate field of Staff is private. You should also assume the Staff class has a getPayRate() method which returns an int (representing the pay in cents); this method is package-private. You should also assume that increasePay() takes two arguments: a reference variable of type Staff (to indicate who is getting a payrise), and a double (to indicate the magnitude of the payrise, in dollars). Assume that the pay rate starts a $0 / hr.

```
Tutor tut = new Tutor();
Administrator admin = new Administrator();
```

```
// Note: these cases must be declared in the administrator package

@Test
public void testZeroPayrise() {
  admin.increasePay(tut, 0.0 );
  assertEquals("Tutor on 0 pay gets 0 payrise",
    tut.getPayRate(), 0);
}

@Test
public void testInsultinglySmallPayrise() {
  admin.increasePay(tut, 0.01 );
  assertEquals("Tutor on 0 pay gets 1-cent payrise",
    tut.getPayRate(), 0.0, 0.01 );
}

@Test
public void testNegativePay() {
  admin.increasePay(tut, -(double)tut.getPayRate()-1.0 );
  assertTrue("Tutor gets huge cut in pay but can't be paid a",
    + "negative amount" + tut.getPayRate() >= 0 );
}
```

9) Minimum wage in New Zealand is $14.75 / hr. Write a test to ensure that the increasePay method can never cause the pay to go below 14.75.

```
Tutor tut = new Tutor();
Administrator admin = new Administrator();
admin.increasePay(tut, 14.75);
```

```
@Test
public void testMinimumWage() {
  admin.increasePay(tut, -0.01 );
  assertTrue("Minimum wage check", tut.getPayRate() >= 14.75);
}
// note that the testNegativePay() method should be deleted
// or revised, because $0.00 is no longer an important boundary
```

10) What is the key difference between black box testing and white box testing

Black box testing is solely concerned with checking if the program functions according to specification. The tester has no access to, nor knowledge of, the internal workings of the software.

Whereas in white box testing the tester has access to, and knowledge of, the inner workings of the software. The tester often focusses on testing branch and loop conditions, to determine whether the program handles all boundary conditions properly (e.g. not exiting a loop 'too soon' or 'too late').

Theme D: application-level concurrent programming

Questions 19, 20, and 21 from the s2 2013 final exam (on the next page). Note that question 22 would not be suitable for this year's examination unless additional information was provided on the invokeLater() and invokeAndWait() methods because these were not covered in this year's lectures or assignments.

19. Consider the following Java code fragment.

```
private int x = 0;
synchronized int incX() {
  return(x++);
}
```

Which of the following is **true**?

a. If two threads attempt to invoke `incX()` simultaneously, both will eventually succeed and the value of `x` may be increased by either 1 or 2.
b. If two threads attempt to invoke `incX()` simultaneously, both will eventually succeed and the value of `x` will be increased by 2.
c. If two threads attempt to invoke `incX()` simultaneously, both will eventually succeed and the value of `x` will be increased by 1.
d. If two threads attempt to invoke `incX()` simultaneously, a `SynchronizationException` will be thrown.

h

(1.5 marks)

20. Which of the following could be observed in a deadlocked Swing application with two workers?

a. The Event Dispatch Thread (EDT) is waiting for a GUI event, and both workers are waiting.
b. The Event Dispatch Thread (EDT) is waiting for a GUI event, one worker is running, and the other worker is waiting.
c. The Event Dispatch Thread (EDT) is waiting for a GUI event, and both workers are running.
d. Any of the above.

a

(1.5 marks)

21. If multiple variables are updated in the body of a `synchronized` method, these changes are

a. Not necessarily atomic, but always visible to all other threads
b. Always atomic, but not necessarily visible to all other threads
c. Always atomic, and always visible to all other threads
d. Not necessarily atomic, and not necessarily visible to all other threads

c

(1.5 marks)

22. In a Swing application, performance problems are likely to arise if

a. The Event Dispatch Thread (EDT) calls `InvokeLater()`
b. The Event Dispatch Thread (EDT) calls `InvokeAndWait()`
c. An initial thread calls `InvokeLater()`
d. An initial thread calls `InvokeAndWait()`

(1.5 marks)