

S266/304

DUBLIN INSTITUTE OF TECHNOLOGY
KEVIN STREET, DUBLIN 8

DIPLOMA IN COMPUTER SCIENCE

YEAR 3

SUMMER EXAMINATION 2004

PROGRAMMING

MR. C. O'LEARY
DR. B. O'SHEA
DR. C. MULVIHILL

20TH MAY 2004, 2:30 – 5:30

ATTEMPT ANY **FOUR** QUESTIONS

ALL QUESTIONS CARRY EQUAL MARKS.

EXACT SYNTAX NOT REQUIRED

1. (a) Discuss fully the differences between *object oriented languages* such as Java, and *procedural languages* such as C. **(5 marks)**

(b) Explain fully how the Java environment supports the philosophy of “*write once, run anywhere*”. **(5 marks)**

(c) Using any example of your choice, show the differences between a Java application and a Java applet. In your answer you should demonstrate how applets can interact with their environment, and the restrictions that apply to this interaction. You must use sample code in your answer. **(8 marks)**

(d) When Java code is executed, exceptions can be raised in the event of an error that has taken place within the code. Using an example and sample code, show how it is possible to handle this exception.

You should also provide code that demonstrates how it is possible to throw exceptions as part of your own code. **(7 marks)**

2. (a) Explain the difference between *instance members* and *class members*, in relation to classes in Java. Use examples and sample code to illustrate your answer. **(3 marks)**

(b) You are required to develop the code to manage a stock room for a company. The company has many different types of product in the stock room. Each product has a *name*, a *price* and a *description*. In addition, all product prices are subject to a *VAT rate* of 20% which is added to the base price. At any point in time, each product will have a certain stock level, which can be increased or decreased.

Using the object oriented principles of *encapsulation* and *information hiding*, design the `Product` class that will contain all the state and behaviour required. Provide a short discussion of how your class design supports encapsulation and information hiding. **(8 marks)**

- (c) The company introduced in part (b) above stores all its products in a stock room. Create a class (named `StockRoom`) that will represent this stock room. This class should include methods to allow for
- i. new products to be added to the stock room,
 - ii. product stock levels to be decreased (when sold),
 - iii. product stock levels to be increased (when purchased),
 - iv. a list of all products to be printed out,
 - v. the value of all stock to be calculated (including VAT).

State any assumptions you made in designing your class. In addition, provide a short discussion of how you designed and implemented your class.

(10 marks)

- (d) Give code for a program control class that could be used to instantiate a `StockRoom` object, add `Product` objects and call each of the methods you created in part (c) above.

(4 marks)

3.



- (a) Provide the Java code necessary to build the GUI shown above. You should describe all your code, using diagrams where necessary to explain how components are positioned relative to each other.

(12 marks)

- (b) Show how the following event handling code could be added to your application:
- i. When the *Fahrenheit to Celsius* box or the *Celsius to Fahrenheit* box is clicked, code is executed that ensures that only one of the two boxes is selected.

(Note: A `MouseListener` has the following event handlers: `mouseClicked(...)`, `mouseEntered(...)`, `mouseExited(...)`, `mousePressed(...)`, `mouseReleased(...)`. Also, a `Checkbox` has the following methods which allow you to check if it is selected, and also to select/unselect it: `getState()`, `setState(boolean state)`)

(5 marks)

- ii. When the *Calculate* button is clicked, the value that has been entered in the first textbox is taken, and is converted to the either Celsius or Fahrenheit, depending on which checkbox has been selected.

Note:

The formula for converting from Celsius to Fahrenheit is:

$$F = ((C / 5) * 9) + 32$$

The formula for converting from Fahrenheit to Celsius is:

$$C = ((F - 32) / 9) * 5$$

Where $F = \text{Degrees Fahrenheit}$ and $C = \text{Degrees Celsius}$

(8 marks)

4. (a) Define the following terms, and using examples and sample code, show how they can be implemented (if possible), in Java.
- i. *inheritance*
 - ii. *multiple inheritance*
 - iii. *abstract classes*
 - iv. *interfaces*
 - v. *superconstruction*
 - vi. *overloading*
 - vii. *overriding*

In your answer, you should discuss why each of these features is useful in an object oriented language.

(14 marks)

- (b) Identify the differences between *primitive types* and *reference types* in Java. You should use sample code to aid your answer.

(3 marks)

(c) Java provides all the usual unary and binary operators, such as +, -, *, ==, % etc. Java also provides a new operator, `instanceof` which can be used for checking reference types prior to casting.

- i. What is meant by the term *casting*?
- ii. Discuss, using examples, how the `instanceof` operator can be used.
- iii. Explain why the following statement will always be true, regardless of the type of `x`:

```
x instanceof java.lang.Object
```

(8 marks)

5. (a) Distinguish fully between the following access modifiers, as applied to class members

- i. `private`
- ii. `public`
- iii. `protected`
- iv. *default (absence of modifier)*

(4 marks)

(b) When one object invokes a method on another object, it is said to “send a message” to that object. The components of this message are the *name of the object*, the *name of the method* and the *arguments to this method*.

Object oriented languages support a particular property that allows objects of the same type to respond differently to the same message.

Identify this property, and discuss in full how it is possible in the Java language.

You should use examples to assist you in your answer.

(9 marks)

(c) The *Command* design pattern requires the property identified in part (b) above.

- i. Explain what is meant by the term *design pattern*, and why design patterns are useful for programmers.
- ii. Describe the *Command* design pattern, identifying situations where it should be used.
- iii. Using sample code, demonstrate how the *Command* design pattern can be implemented in Java

(12 marks)

6. The standard `java.net` package provides a number of classes that can be used to allow distributed objects interact using well known protocols such as TCP, UDP and HTTP.

You are required to implement a distributed application using your choice of these classes. Your application will be comprised of the following components:

- i. A JSP page, named `equals.jsp`, that will accept two parameters over HTTP. Both parameters, named `arg1` and `arg2` represent strings of text. The JSP page will check to see if both strings are equal, and will return a string containing either YES or NO, indicating whether or not the strings are equal.
 - ii. A client, named `Client.java`, that will allow the user to provide two strings as command line arguments. Both strings will then be sent to the JSP page, using HTTP, and the result will be retrieved and displayed.
- (a) Explain what is meant by the term *package*, as used in the above description. In your answer you should demonstrate, using sample code where appropriate, how to create your own packages, and how to import existing packages into an application.
- (5 marks)**
- (b) Provide the JSP code necessary for the above application. Explain each line of your code.
- (5 marks)**
- (c) Provide the code for a class named `Client.java` that will supply all the functionality required for the client, using HTTP GET as the communications protocol. Explain each line of your code.
- (9 marks)**
- (d) Show what changes are necessary to allow the client to communicate with the JSP page using HTTP POST. Explain each line of your code.
- (6 marks)**