



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Examination Answer Books

12 Pages

Instructions to Candidates

1. Please give the information requested in the boxes below.
2. Enter your name in the space provided to the right, fold over and seal down (using the label provided) to hide your name.
3. The answers must be legibly written. The Examiners can either ignore illegible script or have it typed and the expense of typing charged to the candidate.
4. Write your answers in ink.
5. Begin each answer on a fresh page. Do all rough work in the answer book and cross it through.
6. No part of this book is to be torn out; it must be handed in complete.
7. If more books than one are used, fasten them together using the tags available from the invigilator(s).
8. **Every book must be handed in. In no circumstances may answer books, used or unused, be taken out of the examination venue.**

Complete, fold and seal

(see note 2 below)

SURNAME: Rees

FORENAME(S): Gwynndaf
Charles

1	Student Number	1444 78
2	Title of Examination Paper	Application Design and development
3	Module Code	M5X01025
4	Date and Time of Examination	14/5/2018 9:00
5	Examination Venue	Pembrokeshire College
6	Seat Number	2
7	Number of Answer Books handed in	1

List the numbers of the questions you have attempted	Office Use
1a	9
1b	4
1c	4
1d	4
2a	5
2b	7
2c	7
3a	6
3b	3
3c	7
Office Use	56

a)

i) Attributes are class specific variables that are used to hold data for use in the method. Encapsulation hides these attributes and their data, keeping them private, and manipulating them through ^{method} get/set. An example would be a dog class with a fur color attribute which can be set ^{through a method}.

ii) An object is an instance of a class. The class is a template for an object, defining the data it can contain and its functionality. An example of this could be an animal class with

~~attributes such as 'sound' and a method of 'play sound'.~~ ~~Inheritance is the creation of a parent-child relationship.~~

attributes such as 'sound' and a method of 'play sound'. For an object of this class, a sound is set and can be played through the 'play sound' method.

iii) Inheritance is the creation of a parent-child relationship. This creates a hierarchy and as you move down the hierarchy from the parent classes to the children, each class adds more specific attributes and/or behaviours. ~~Because~~ This is called specialisation. ~~This relationship~~ An example of this is a 'Dog' class being a child of the 'Animal' class as a Dog is an Animal and, therefore, should inherit attributes general to all animals.

b) The example defines two classes, a Dog class, which defines a ^{public} function, Bark, which writes the word 'Bark' to the console; and a House class, which creates a Dog object as the 'guardDog' attribute and then defines a private method, called 'changeDog', ~~and two public methods, those which change~~.

^{using constructor} the ~~dog~~ ^{'guardDog'} attribute to a different Dog object passed in as a parameter. This method is used to set the value of the ~~dog~~ ^{'guardDog'} attribute on creation of a House object. The class also has a public method, 'Bang Door', which calls the 'guardDog' attributes 'Bark' function which was defined in the Dog class. (4)

- C) ^{constructor} The cat function of the cat class passes a string ~~called~~ ^{called} 'theCatName' but sets the 'name' attributes value to an unknown 'name' variable instead of 'theCatName'. The Owner class has a method called 'owner' which ~~passes~~ ^{has} an ~~extra parameter with the~~ owner object called cat as its parameter. It then sets the value of the 'cat' attribute, which is a Cat object, to this value. These ~~the~~ data types are different. The 'GreetCat' method writes a string to the console, ~~and does~~ but it attempts to call the cat classes 'name' attribute which is private. This should have used the ~~the~~ 'getName' method. The constructor for the cat class should have a return type, even void. (4)

d)

```
Dog dog = new Dog();
```

```
House house = new House(dog);
```

```
dog.Bark();
```

//dog object barks

```
house.BangDoor();
```

//House 'guard dog' barks

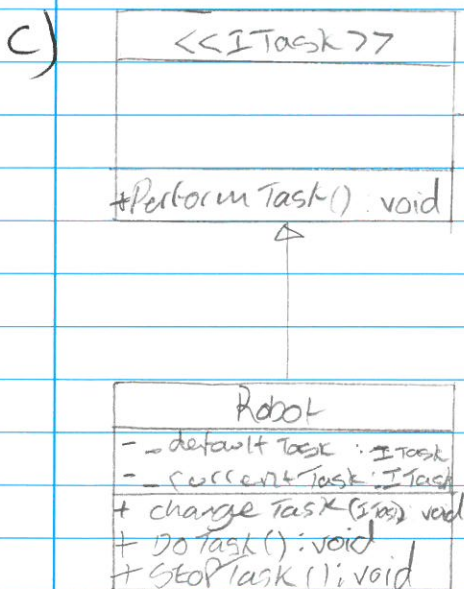
4

2) An instance of the House class has one owner (an instance of the Owner class). An owner ^{depends upon} ~~may have~~ a car. A House may have many parts. ~~the part~~ A part may be a 'window' or 'door' which are children of part. The part class allows the opening and closing of ~~the~~ its child objects, 'door' and 'window'. The part class also specifies the material of its children.

(5)

b) The Robot class uses two attributes, both of which are instances of the ITask interface. This abstraction allows the class to use the performTask function defined elsewhere. This type of abstraction is an instance of the Strategy pattern.

(7)



Constructor?

(8)

3)

a) The Single Responsibility principle states that Every module or class should have responsibility over a single part of the functionality provided by the software, and that responsibility should be entirely encapsulated by the class. This essentially means that a class or module only have one reason to be changed. The responsibility aspect is the work or action each part of the system is assigned to do. Too much responsibility leads to coupling, which is the level of detail one part of the system knows about another. Coupling complicates code changes and adhering to the single responsibility principle helps against this problem by simplifying the changes.

(6)

b) The system allows a user to make an order, either a 'mobile phone app' order or a 'web' order. ~~There is a~~ The user can then view the orders (Display orders), or confirm the order, which ~~then~~ then takes payment.

(3)

c) Black box testing is data driven whereas white box is logic driven. Black box requires the tester to be unconcerned with the internal behaviour of the system, instead concentrating on finding circumstances where the program does not behave according to specification. White box testing

is entirely concerned with the inner logic of the program, deriving test data to test its handling of it. White box testing risks biased tests from developers who tailor tests to the project's function, not ~~the~~ what it is required to do. A combination of both types of testing should be used although it's important to not over test some components and under test others with black box testing, and not to miss key end-user scenarios. Gray box testing where tests are designed from a customer focused perspective should also be considered.

(7)