



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: Price

FORENAME(S): David

1	Student Number	P131183
2	Title of Examination Paper	Application Design and Development
3	Module Code	MSX01025
4	Date and Time of Examination	9:00am Monday 14 May 2018
5	Examination Venue	Townhall Gym
6	Seat Number	G092
7	Number of Answer Books handed in	1

List the numbers of the questions you have attempted	Office Use
1ai	6 1/2 ✓
1aii	
1aiii	
1b	
1c	10 1/2 ✓
1d	
2a	
2b	
2c	17 ✓
3a	
3b	
3c	
	34/75
	GT 1/6/18
Total	45%

(Gweler yr ochr arall am y fersiwn Cymraeg)

2ai) Attributes ^{is} ~~are~~ data that is relevant to the class,
~~the~~ data has many different types to
 determine what type of attribute it is ✓
 eg. int - num 1
 date - height

1 1/2



Encapsulation is grouped data together

2aii) Object relationship to class is that an object
 is created from the class in which it
 contains a unique value for each attribute

2



2aiii) ~~Inheritance is created from a parent-child relationship~~

Example of an object:

Class car

objects of car → BMW, Aston martin, mini, mercedes etc

2aiii) Inheritance is created from a parent-child relationship ✓
 it is used to avoid duplication of the code.

1 1/2



Parent class Animal

Child class Dog

Animal myAnimal = new Dog();
 myAnimal.makeNoise();

1b) The code in appendix 1 shows parent class as
 house and child class as Dog, ✗
 The code is showing that the parent class can
 change the child class ✓ at any point with the
 methods and that the methods in the



1b) In the parent class house are inherited from the dog class as it stops duplication of code to have methods only in the child class. ✗

The parent class allows data encapsulation, makes the dog class and refers to it in the code ✓

1c) The four mistakes are:

1. the cat class name attribute is set to private which ~~isn't~~ contained within that class only, should be made public. ✗

2. no method to set the name only to getName ✗

3. the constructor in cat class "theCatName" isn't used ✓

4. Owner ~~to~~ class isn't inheriting from cat class, no syntax is used class Owner : cat, to retrieve the name of the cat as it was initially set private so need the inheritance there. ✗

1d) class mainmap

```

At the end
class Dog
{
    public void Bark ()
    {
        console.WriteLine("Bark");
    }
}
  
```

```

1d) class House
{
    private dog - guardDog;

    public House (Dog dog)
    {
        Change Dog (dog);
    }

    public void BangDoor()
    {
        -guardDog = Bark();
    }

    private void ChangeDog (Dog dog)
    {
        -guardDog = dog;
    }
}

```

main.c

```

static void
{
    House myHouse = new Dog();
    myHouse.BangDoor
}

```

```

Dog myDog = new Dog(); ✓
myDog.BangDoor(); ✗

```

1/2

✓

2a)

House is related to both owner and parts, owner is related to car with attributes such public DriveCar();, this implies the owner is capable of driving the car.

The owner needs to drive the car - due to the relationship between House and parts, also also parts and window and Door.

Window Parts has an attribute called public open() and public close();, this is for the child class window ~~a window~~ and door which can both be opened or closed.

The attribute in parts is public string Part(); which implies parts are needed,

The House has an attribute called private parts(), this is related to parts, therefore related to window and door,

The car is needed to grab the parts from the data collected from the child classes of Part which are window and door. House displays the parts needed, the other classes retrieve them.

2b)

Gang of four

Abstraction is used in OOP ~~to~~ is for

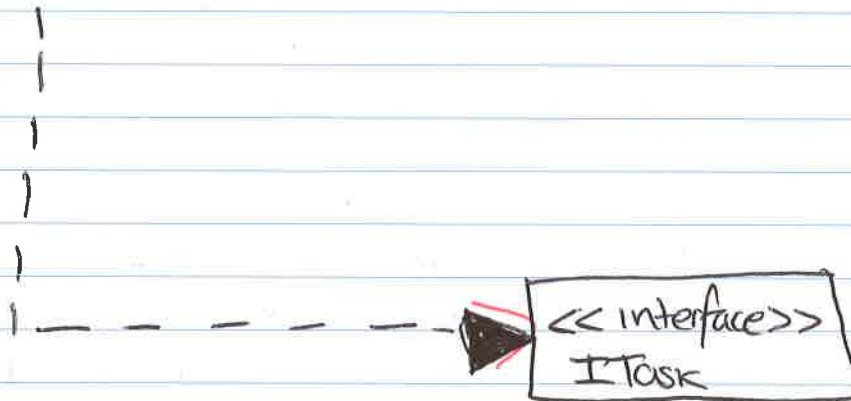
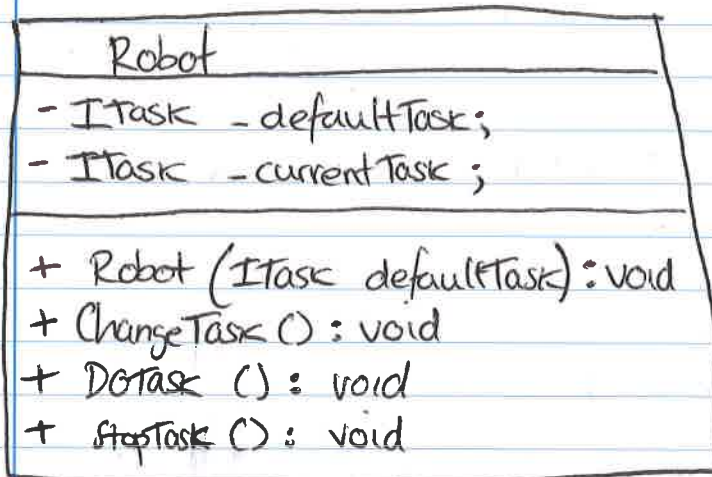
- Maintainability
- Extensibility
- Testability

It leads to more ~~read~~ readable and flexible

code.

The abstraction used in the example code Appendix 4, is built for the purposes stated on the previous page.

2c)



6 1/2



Perform Task() : void
10 1/2

3a) The single responsibility principle (SRP) is where a class should

- fulfil only one responsibility ✓
- only one reason to change ✓

Avoid changes to classes, modifying code increases the likelihood of errors

- Do one thing
- Do that thing only
- Do that thing well



3a)

If a class has more than one responsibility can lead to increased coupling

- The responsibilities may rely on each other
- Changing one responsibility may damage the other code

If a two subsystem is introduced using different methods or parts of a class, dividing responsibilities this way

- Avoids of a monolithic class design
- Improves readability
- Reduces code complexity
- Reduces coupling
- Improves testability
- High cohesion

3b)

In this diagram it shows the process of a user purchasing a mobile phone app

The user relationship is with the 'make order' , the user makes order, the subclasses of 'make order' are 'web order form' and 'mobile phone app order', these two classes are related to make order so after they run successfully, data is then passed to the user

User can display order to see if the correct order has been made, after that is successful

The order is confirmed, the interface will include a payment, the payment is run and then the user is notified payment successful.

3c) SOCK

- S - simple applications are easier to test
- O - Observable structure to show if a test is passed or failed
- C - Control area set thresholds
- K - knowledge, name documentation of the test

White box testing

- logic driven testing
- used to test low level libraries/functions ✓

The risk is easy to become biased

Testing using the debugger

- Data flow
 - Trace specific data through the code
 - Where does it change?
- Error forcing
 - Force variables to specific values and how your code handles it

Verification - check consistency of an implementation

Validation - Check it meets users needs

3c) Black box testing

- Data driven / input / output driven testing
- finding circumstances where the code does not behave accordingly

Testing Strategy

- Test to pass - test software minimally
- first test to try
 - Not trying to break the code

- Test to fail - Be sneaky, force bugs out
- Performed after test to pass
 - Probe weaknesses

Input Equivalence Partitioning

Exhaustive input testing

- find all errors in the program use every possible input condition

Range - contiguous set of data

Group - items can be handled together

Unique - Data handled differently

Specific - must or must not be present

The use of either ~~test~~ testing can be used in software development, there are pros and cons for both as listed.