Programming Paradigm

Q- What is meant by Programming Paradigm?

- A programming style or classification or the features that the programming language has.

① Low-Level ② Imperative ③ Object-Oriented ④ Declarative

Assembly Language (Low-Level)

- CASE 1: Binary Multiplication

128	64	32	16	8	4	2	1	
0	0	1	0	0	0	1	0	=> 34

Q- Multiply 34 by 2

2 = 2^1 ↪ Left shift #1

	128	64	32	16	8	4	2	1	
0	0	1	0	0	0	1	0	0	=> 68 → 34x2

Q- Multiply 34 by 4

$4 = 2^2$ → Left shift by #2

		128	64	32	16	8	4	2	1	
0	0	1	0	0	0	1	0	0	0	→ 136

- We cannot multiply with numbers which are not included in the table of 2 e.g: 9 and 5
- For divide use Logical shift right

Increment In a Variable

- LDD Count
- INC ACC
- STO count

Questions

File 1:

Q22 , Q31

File 2:

Q9, Q15, Q16, Q18,

Q22, Q25, Q35

* Revise bit-manipulation from AS.

A programmer writes a program that multiplies two numbers together and outputs the result. The numbers are stored as NUMONE and NUMTWO.

The programmer has started to write the program in the following table. The comment column contains explanations for some of the missing program instructions and data.

Complete the program using the given instruction set.

Label	Op code	Operand	Comment
LOOP:	LDD	ANSWER	// load the value from ANSWER
	ADD	NUMONE	// add the value from NUMONE
	Sto	Numone	
	LDD	Count	// load the value from COUNT
X	INC	ACC	// increment the Accumulator
	LDD	Numtwo.	
	CMP	Count	// is NUMTWO = COUNT ?
	JPN	Loop.	// if false, jump to LOOP
	LDD	Answer.	// load the value from ANSWER
	Out		// output ANSWER to the screen
	END.		// end of program
NUMONE:	2		
NUMTWO:	4		
COUNT:	0		
ANSWER:	0		

[9]

(b) A programmer needs a program that counts the number of lower case letters in a string.

The programmer has started to write the program in the following table. The comment column contains explanations for the missing program instructions.

Complete the program using the given instruction set. A copy of the instruction set is provided on the opposite page.

Label	Instruction		Comment
	Op code	Operand	
	LDR	#0	// initialise Index Register to 0
START:	LDX	STRING	// load the next value from the STRING
	AND	MASK	// perform bitwise AND operation with MASK
	CMP	MASK	// check if result is equal to MASK
	JPN	UPPER	// if FALSE, jump to UPPER
	LDD	Count	
	Inc	ACC	// increment COUNT
	sto	Count	
UPPER:	INC	IX	// increment the Index Register
	LDD	LENGTH	
	DEC	ACC	// decrement LENGTH
	sto	LENGTH	
	CMP	#0	// is LENGTH = 0 ?
	JPN	START	// if FALSE, jump to START
	END		// end program
MASK:	B00100000		// if bit 5 is 1, letter is lower case
COUNT:	0		
LENGTH:	5		
STRING:	B01001000		// ASCII code for 'H'
	B01100001		// ASCII code for 'a'
	B01110000		// ASCII code for 'p'
	B01110000		// ASCII code for 'p'
	B01011001		// ASCII code for 'Y'

[8]

(b) The programmer now starts to write a program that:

- converts a positive integer, stored at address NUMBER1, into its negative equivalent in two's complement form
- stores the result at address NUMBER2

Complete the following program. Use op codes from the given instruction set.
Show the value stored in NUMBER2.

Label	Op code	Operand	Comment
START:	LDD	Number1	
	XOR	MASK	// convert to one's complement
	INC	ACC	// convert to two's complement
	Sto	Number 2	
	END		
MASK:	B 11111111		// show value of mask in binary here
NUMBER1:	B00000101		// positive integer
NUMBER2:	B11111011		// negative equivalent

[6]

(a) A programmer needs a program that multiplies a binary number by 4.

The programmer has started to write the program in the following table. The comment column contains explanations for the missing program instructions.

Write the program using the given instruction set.

Label	Instruction		Comment
	Op code	Operand	
	LDD	NUMBER	// load contents of NUMBER
	LSL	#2	// perform shift to multiply by 4
	Sto	NUMBER	// store contents of ACC in NUMBER
	END		// end program
NUMBER:	B00110110		

Declarative Programming

· Declarative Programming is used to extract knowledge by the use of queries from a situation with known facts and rules.

↓ Things that are known

↘ Relationship b/w facts

Examples Of Facts:

students (aliwajid) ⊙

students (shuja).

students (ahmar).

students (sheriyar).

students (taha).

students (hashim).

students (abdullah).

teacher (taha, emad).

· Always remember the Full stop at the end of a fact

· Always write in small letters, the info like the one on left.

· Do not use Capital letter as the first letter, unless it is a variable e.g: students (Name)

↳ used as a variable

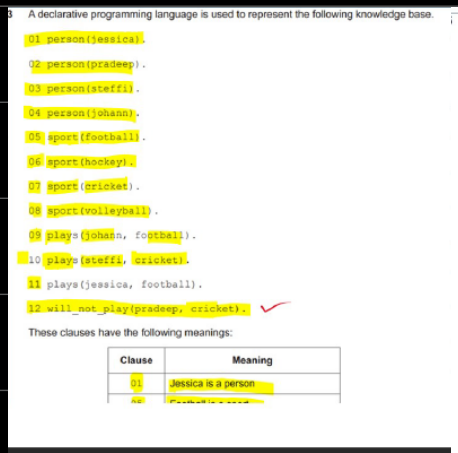
Name = aliwajid, shuja .... (order is important, starting from top and going down to bottom)

- Facts are written to create a knowledge base, then this knowledge can be used.

Questions

File 1:

3, 11, 35, 41, 45, 49

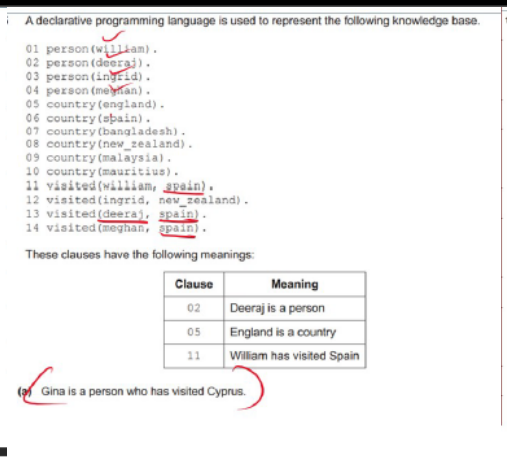


3 A declarative programming language is used to represent the following knowledge base.

```
01 person(jessica).
02 person(pradeep).
03 person(steffi).
04 person(johann).
05 sport(football).
06 sport(hockey).
07 sport(cricket).
08 sport(volleyball).
09 plays(johann, football).
10 plays(steffi, cricket).
11 plays(jessica, football).
12 will_not_play(pradeep, cricket).
```

These clauses have the following meanings:

Clause	Meaning
01	Jessica is a person



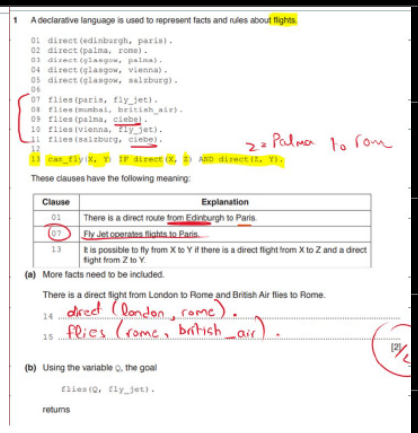
A declarative programming language is used to represent the following knowledge base.

```
01 person(william).
02 person(deeraj).
03 person(ingrid).
04 person(meghan).
05 country(england).
06 country(spain).
07 country(bangladesh).
08 country(new_zealand).
09 country(malaysia).
10 country(mauritius).
11 visited(william, spain).
12 visited(ingrid, new_zealand).
13 visited(deeraj, spain).
14 visited(meghan, spain).
```

These clauses have the following meanings:

Clause	Meaning
02	Deeraj is a person
05	England is a country
11	William has visited Spain

(a) Gina is a person who has visited Cyprus.



1 A declarative language is used to represent facts and rules about flights.

```
01 direct(edinburgh, paris).
02 direct(palma, rome).
03 direct(glasgow, palma).
04 direct(glasgow, vienna).
05 direct(glasgow, salzburg).
06
07 flies(paris, fly_jet).
08 flies(mumbai, british_air).
09 flies(palma, ciebs).
10 flies(vienna, fly_jet).
11 flies(salzburg, ciebe).
12
13 can_fly(X, Y) IF direct(X, Z) AND direct(Z, Y).
```

Z = Palma to rome

These clauses have the following meaning:

Clause	Explanation
01	There is a direct route from Edinburgh to Paris.
07	Fly Jet operates flights to Paris.
13	It is possible to fly from X to Y if there is a direct flight from X to Z and a direct flight from Z to Y.

(a) More facts need to be included.

There is a direct flight from London to Rome and British Air flies to Rome.

14 direct (london, rome).
15 flies (rome, british_air).

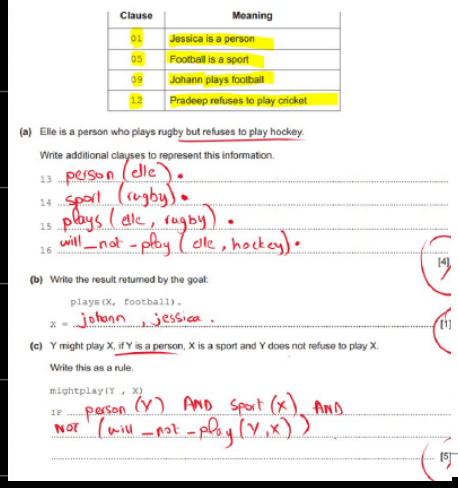
(b) Using the variable Q, the goal

```
flies(Q, fly_jet).
```

returns

File 2:

2, 8, 13, 30, 33, 36



Clause	Meaning
01	Jessica is a person
05	Football is a sport
09	Johann plays football
12	Pradeep refuses to play cricket

(a) Elle is a person who plays rugby but refuses to play hockey.

Write additional clauses to represent this information.

13 person (elle).
14 sport (rugby).
15 plays (elle, rugby).
16 will_not_play (elle, hockey).

(b) Write the result returned by the goal:

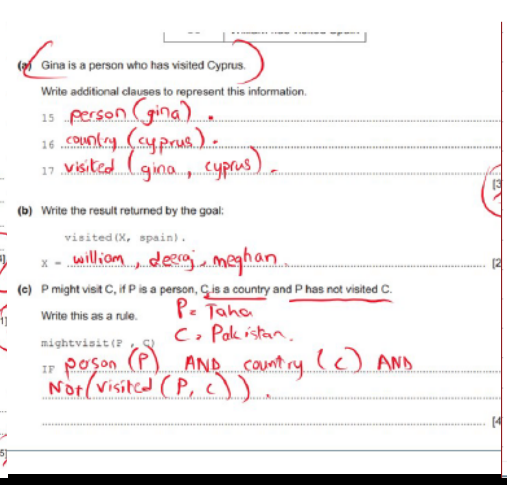
```
plays(X, football).
```

X = johann, jessica.

(c) Y might play X, if Y is a person, X is a sport and Y does not refuse to play X.

Write this as a rule.

```
mightplay(Y , X)
IF person(Y) AND sport(X) AND
NOT (will_not_play(Y,X))
```



(a) Gina is a person who has visited Cyprus.

Write additional clauses to represent this information.

15 person (gina).
16 country (cyprus).
17 visited (gina, cyprus).

(b) Write the result returned by the goal:

```
visited(X, spain).
```

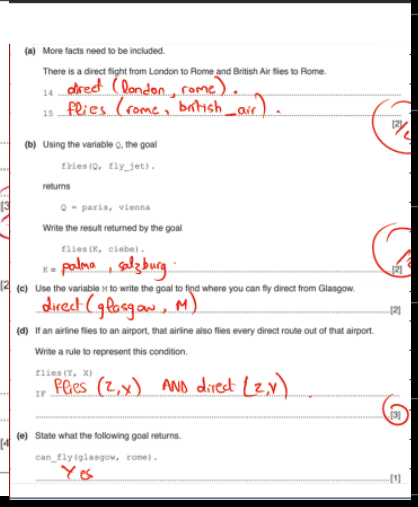
X = william, deeraj, meghan

(c) P might visit C, if P is a person, C is a country and P has not visited C.

Write this as a rule.

P = Taha
C = Pakistan.

```
mightvisit(P , C)
IF person(P) AND country(C) AND
Not(visited(P, C)).
```



(a) More facts need to be included.

There is a direct flight from London to Rome and British Air flies to Rome.

14 direct (london, rome).
15 flies (rome, british_air).

(b) Using the variable Q, the goal

```
flies(Q, fly_jet).
```

returns

Q = paris, vienna

Write the result returned by the goal

```
flies(K, ciebe).
```

K = palma, salzburg

(c) Use the variable K to write the goal to find where you can fly direct from Glasgow.

direct (glasgow, M)

(d) If an airline flies to an airport, that airline also flies every direct route out of that airport.

Write a rule to represent this condition.

```
flies(Y, X)
IF flies(Z,X) AND direct(Z,Y)
```

(e) State what the following goal returns.

```
can_fly(glasgow, rome).
```

Yes

Object-Oriented Programming (OOP)

Q- What is meant by OOP?

- A more organized way of programming

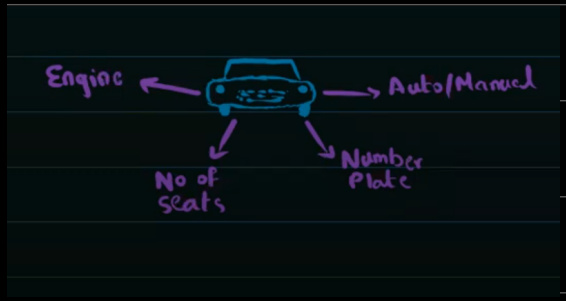
The data and the functions that belong to a single entity can be grouped together in a single object

Courses

Login Form

STUDENT

- Each object holds its specific information

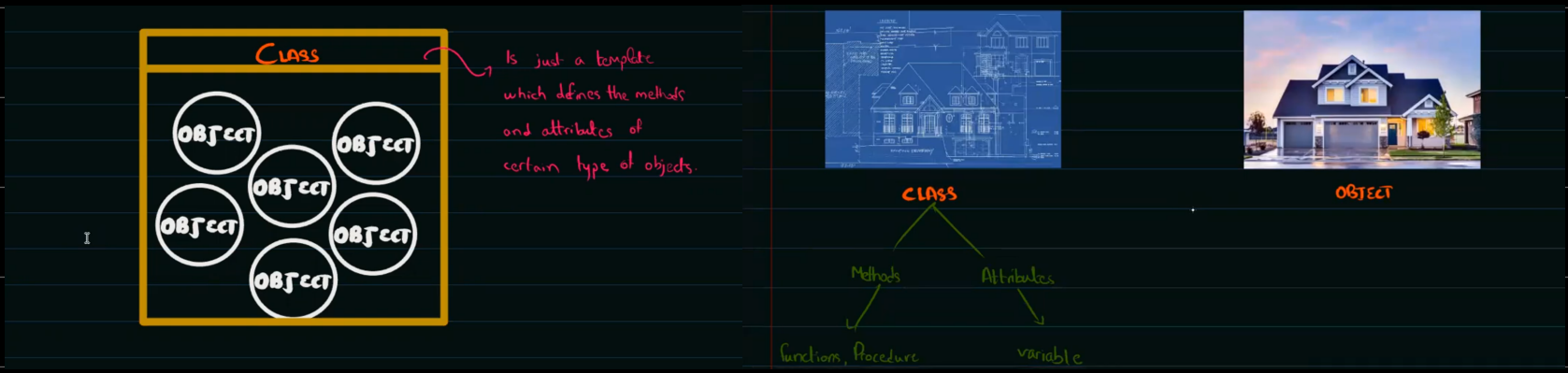


STUDENT → Name, contact, email, age, Father's Name

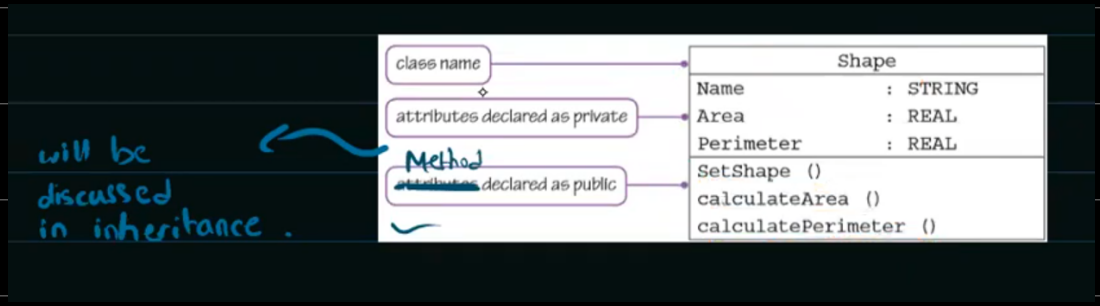
ATTRIBUTES

- An object not only contains attributes but also some actions
- Those actions are basically functions / procedures
- known as Methods

Class And Object



- Putting the data (attributes) and methods together as a single unit (class) is called encapsulation.

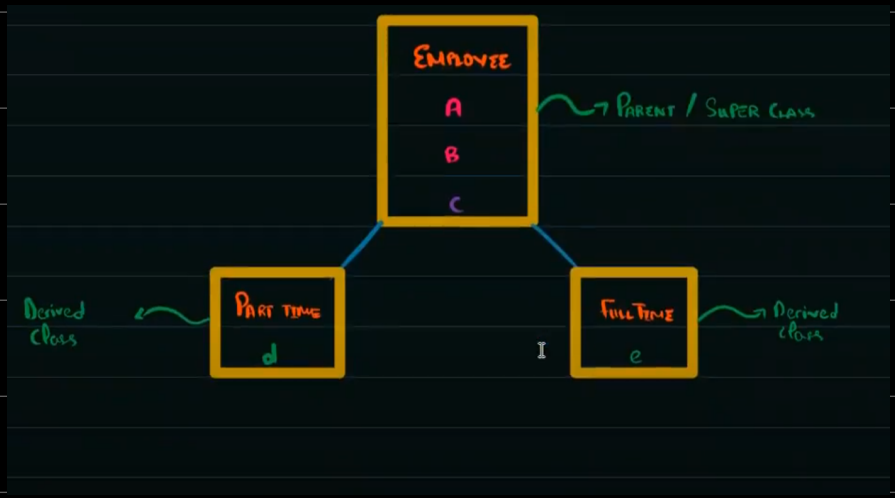


Main Features of OOP

- Inheritence
- Polymorphism
- Containment / aggregation

Inheritance

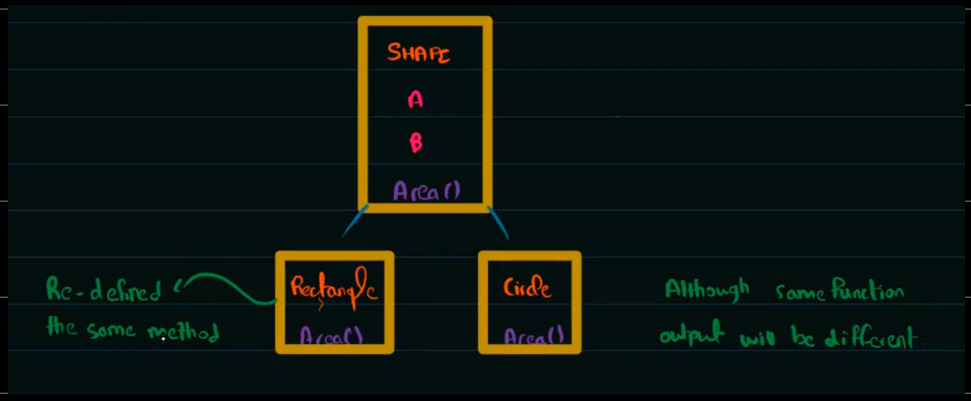
Solution



Q- What is meant by Inheritance?

- The derived class can use the properties from the parent class without re-declaring them
- " " " can use the methods " " " " " " "
- Can extend properties from parent class
- " " the methods from the parent class

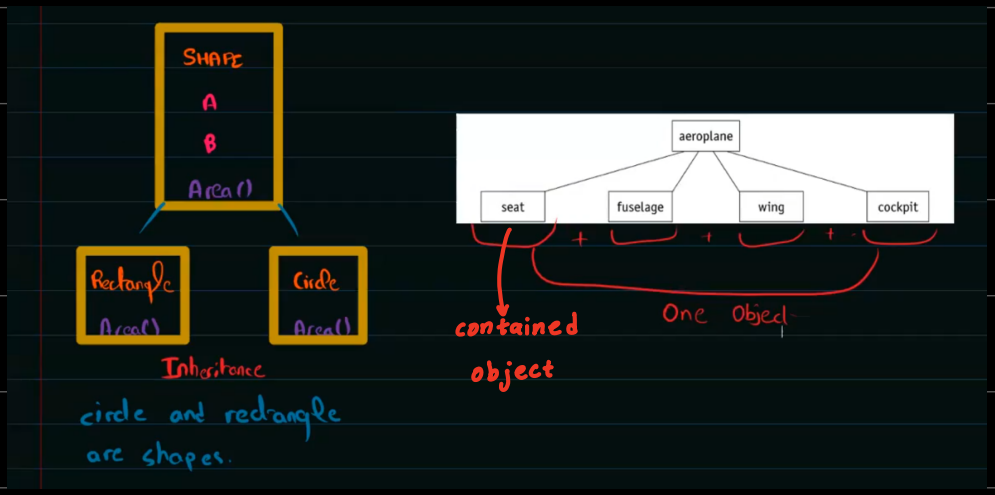
Polymorphism



Q- what is meant by polymorphism?

- Is when methods are redefined for derived classes

Containment



Q- What is meant by containment?

- A class include instance (object) of another class
- Aggregation (formation of a single object from multiple objects)
- The contained object can exist outside of its super class
- Object is destroyed when super class is destroyed.

Q- List Features of OOP

- Encapsulation
- Private / Public
- Inheritence
- Methods
- Properties
- Classes
- Polymorphism
- Containment

Overloading: When a method is defined more than once, so that it can be used in different scenarios.

Python

```
class greeting:
    def hello(self, name = None):
        if name is not None:
            print ("Hello " + name)
        else:
            print ("Hello")
myGreeting = greeting()
myGreeting.hello()
myGreeting.hello("Christopher")
```

method used with no parameters

method used with one parameter

* destructors

Setter: Is a method used to control changes to any variable that is declared within an object

When a variable is declared as private, only the setters declared can be used to make changes

Getter: Is a method that gets the value of a property of an object

Q- Explain how get and set methods are used to support security and integrity?

- Used to change the properties that are set to private.
- by only using get/set methods
- Provides encapsulation
- Prevents accidental change.
- To make sure data is valid
- Hides Data
- The get method allows the data to be returned
- The set method allows the data to be changed.

Pseudocode of Object Oriented Programming

① Defining Class:

```
CLASS Pet
    PRIVATE Name: String
    PUBLIC PROCEDURE NEW(Given name : STRING)
        Name ← Givenname
    END PROCEDURE
END CLASS
```

NEW: keyword for constructor

Constructor

Private: Attribute can only be accessed inside the class

② Derived Class:

```
CLASS Cat INHERITS Pet
    PRIVATE Breed: String
    PUBLIC PROCEDURE NEW(Givenname: STRING, GivenBreed: STRING)
        SUPER.NEW(Givenname)
        Breed ← GivenBreed
```

```
        END PROCEDURE

    END CLASS
```

(3) Setter & Getter:

```
CLASS Pet

    PRIVATE Name: STRING

    PUBLIC PROCEDURE NEW ( GivenName : STRING)
        Name ← GivenName
    END PROCEDURE

    PUBLIC PROCEDURE setName ( N: STRING)
        Name ← N
    END PROCEDURE

    PUBLIC FUNCTION GetName () RETURNS STRING
        RETURN Name
    END FUNCTION
```

Constructor

Setter

Getter

END CLASS

④ Creating an Object :

<object name> ← NEW <Class Name> (Parameter 1, Parameter 2, ...)

e.g: MyCat ← NEW Cat ("kitty", "Persian")

⑤ Calling Methods :

MyCat.GetName()

"kitty" is returned

MyCat.setName("Fred")

MyCat.GetName()

"Fred" is returned

MyCat.Name() ✗

↳ Can not be used outside class because of private declaration. If public, then can be used outside class and will return "Fred"

Q- What is meant by imperative programming paradigm?

- A sequence of steps that change the state of the program
- The steps are in order they should be carried out
- e.g: Procedural / structured programming
- Groups code into self-contained blocks
- which are sub-routines

Q- What is meant by object oriented programming paradigm?

- Creates classes
- as a blueprint for an object
- that can have properties and methods
- That can be private to the class
- Subclasses can inherit from superclasses.
- A subclass can inherit the methods and properties from the super class
- A subclass can change the methods from the superclasss (polymorphism)
- objects can interact with each other.

Q- low-level Programming

- Uses instructions from the computer's basic instruction set
- Assembly code and machine code both use low-level programming language
- Used when program needs to make use of specific addresses and registers in a computer. e.g: writing a printer driver