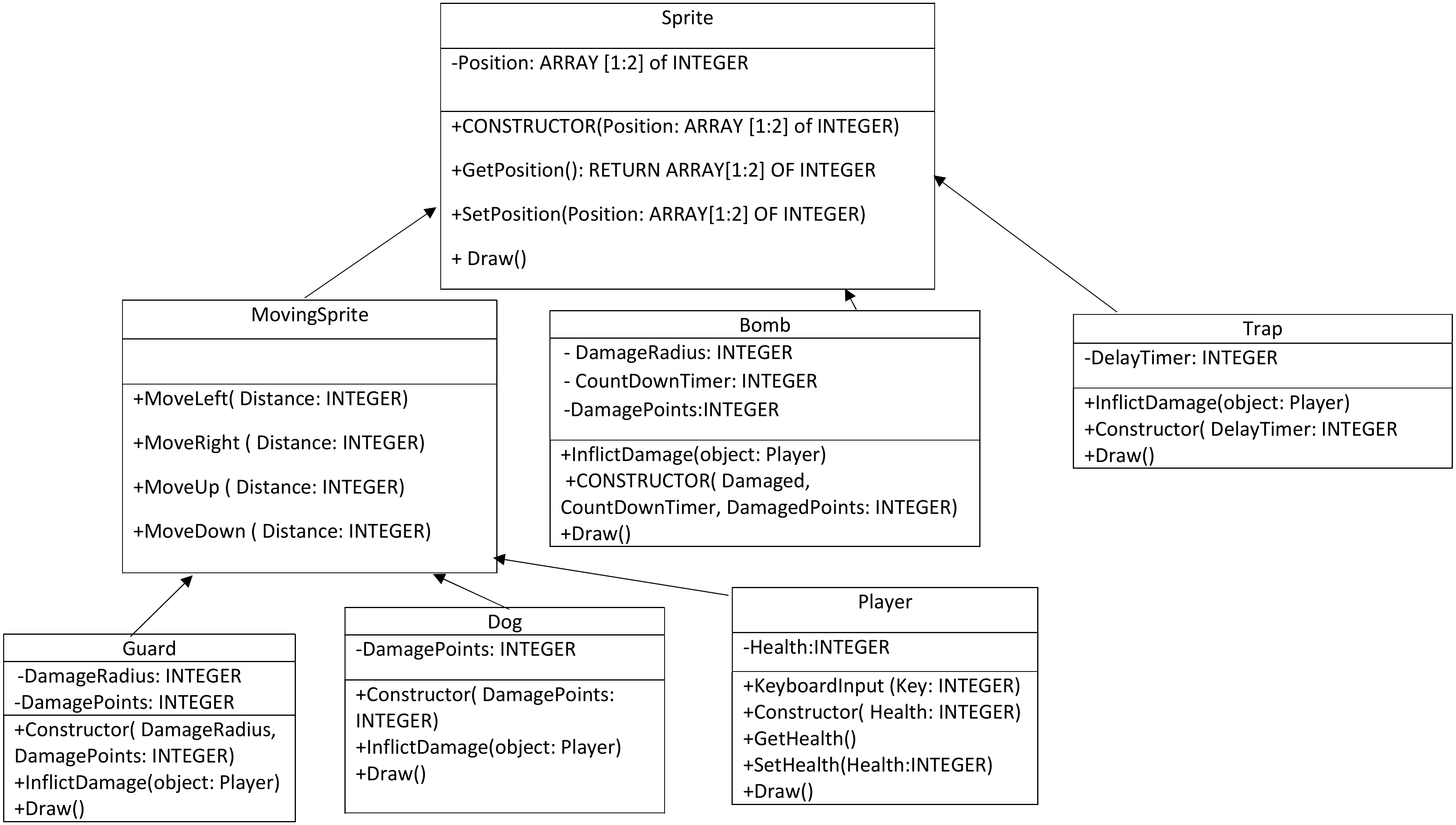
Question 2: a) UML Class Diagrams



Sprite

-Position: ARRAY [1:2] of INTEGER

+CONSTRUCTOR(Position: ARRAY [1:2] of INTEGER)

+GetPosition(): RETURN ARRAY[1:2] OF INTEGER

+SetPosition(Position: ARRAY[1:2] OF INTEGER)

+ Draw()

MovingSprite

+MoveLeft(Distance: INTEGER)

+MoveRight (Distance: INTEGER)

+MoveUp (Distance: INTEGER)

+MoveDown (Distance: INTEGER)

Bomb

- DamageRadius: INTEGER
- CountDownTimer: INTEGER

-DamagePoints:INTEGER

+InflictDamage(object: Player)
+CONSTRUCTOR(Damaged, CountDownTimer, DamagedPoints: INTEGER)
+Draw()

Trap

-DelayTimer: INTEGER

+InflictDamage(object: Player)
+Constructor(DelayTimer: INTEGER
+Draw()

Guard

-DamageRadius: INTEGER
-DamagePoints: INTEGER

+Constructor(DamageRadius, DamagePoints: INTEGER)
+InflictDamage(object: Player)
+Draw()

Dog

-DamagePoints: INTEGER

+Constructor(DamagePoints: INTEGER)
+InflictDamage(object: Player)
+Draw()

Player

-Health:INTEGER

+KeyboardInput (Key: INTEGER)
+Constructor(Health: INTEGER)
+GetHealth()
+SetHealth(Health:INTEGER)
+Draw()

Marking Scheme:

- Identify common game sprite/character: **Sprite class [1]**
- Identify sprite/character that are mobile and immobile: **Moving Sprite, Bomb, Trap Classes [2]**
- Identify the specialised game sprite/character: **Guard, Dog, Player [2]**

b)
- Inheritance allows code re-use [1]
- All characters needs to have aposition (x,y) on the screen [1] OR
- Characters can move , share code for moving [1]

c)
- Polymorphism have different behavior for the same operation [1]
- When the charcaters are rendered on the screen, they are drawn differently with some common basic drawing provided by the base class[1] OR
- When the Bomb,Trap,Guard,Dog needs to InflictDamage they have different damage values to deduct from the Player [1]

Question 1

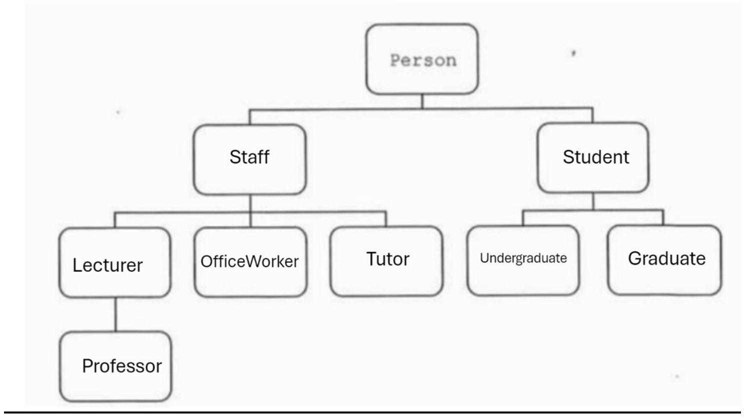
1(a)	The CAR class is a subclass of the VEHICLE class.	
1(b)(i)	Abstraction shows only the essential information (e.g. speed and position of the vehicle) and hides the unnecessary details. This helps to improve programming efficiency and reduce confusion.	
1(b)(ii)	Inheritance allows a subclass to take on properties and methods of the superclass without having to write them all over again. This reduces the amount of editing if something is changed in the superclass and hence, the chance of making a mistake.	

1(c)	The petrol use rate depends on different attributes in each subclass – the number of `Passengers` in the case of `CAR`, and the `Current_Load` in the case of `TRUCK`. Hence, the attribute would need to be written differently for each of the subclasses, and this is an example of polymorphism.	

Question 3

(a) object oriented programming is a programming paradigm where the principles of encapsulation, inheritance and polymorphism is used to implement code. Encapsulation allows the binding of code to its data and hiding the implementation details. Inheritance allows code re-use and Polymorphism allows a derived class to override the default behaviour of its parent class.

(b)



ii) Inheritance allows a subclass to re-use the code of its superclass. By re-using code that has already been written, it allows modifications to be made only at one place only thereby making code maintenance much more efficient.