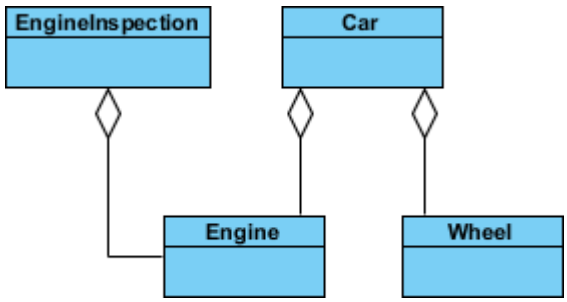
OOPT PYQ 09/2018

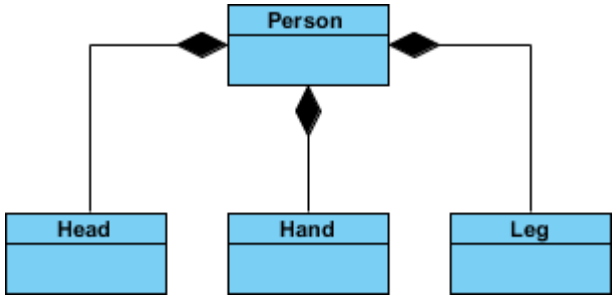
Question 1

Part A

1. Aggregation is a relationship between classes where **one class is a part of another class**. The child class can exist independent of the parent class.

1. 

2. Composition is a **special case of aggregation** where one class belongs to another class exclusively. The child class cannot exist independent of parent class.

1. 

Part B

1. **Encapsulation**: The binding of methods and their data together in a single, cohesive object known as class.
2. **Inheritance**: The creation of classes that are a superset of another class, having all the methods and properties of a more general class.
3. **Polymorphism**: The declaration of functions inside parent classes, and leaving the definitions of functions to the child classes. Every child class can have different method definitions, despite having the same method names.

Part C

1. 3 benefits
    1. **Code reusability.** A method can be written once and be constantly reused.
    2. **Make code more readable.** Methods hides away lower-level details that define an action, allowing a programmer to focus more on design than underlying code.
    3. **Make code easier to debug**. Methods allow a programmer to isolate errors by localizing the errors. Errors in multiple small methods can be easily traced and found compared to finding it through one big program.
2. Method overloading refers to methods that have the same method name, but **different parameter listing**.

```java
public class Point {
    int x;
    int y;
    public Point() {};
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

Question 2

Part A

```java
public class SchoolBag {
    private String id;
    private String material;
    private int volume;
    private double price;
    private static int totalNoOfSchoolBag;

    public SchoolBag() {
        totalNoOfSchoolBag++;
    }

    public SchoolBag(String id, String material, int volume, double price) {
        this.id = id;
        this.material = material;
        this.volume = volume;
        this.price = price;
        SchoolBag.totalNoOfSchoolBag++;
    }

    public void setID(String id){
        this.id = id;
    }

    public void setMaterial(String material) {
        this.material = material;
    }

    public void setVolume(int volume) {
        this.volume = volume;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public String getID() {
        return id;
    }

    public String getMaterial() {
```

```
        return material;
    }

    public int getVolume() {
        return volume;
    }

    public double getPrice() {
        return price;
    }

    public String toString() {
        return "ID: " + id + "\n" +
            "Material: " + material + "\n" +
            "Volume: " + volume + "\n" +
            "Price: RM" + price + "\n";
    }

    public static int getTotalNoOfSchoolBag() {
        return totalNoOfSchoolBag;
    }
}
```

Part B

```
public class SchoolBagDriver {
    public static void main(String args[]) {
        SchoolBag s = new SchoolBag();

        s.setID("B0234");
        s.setMaterial("Canvas");
        s.setVolume(35);
        s.setPrice(55.00);

        SchoolBag s2 = new SchoolBag("B3278", Nylon, 30, 40.90);

        System.out.println(s);
        System.out.println(s2);

        System.out.println("Total no. of school bags: " +
SchoolBag.getTotalNoOfSchoolBag());
    }
}
```

Question 3

Part A

```
public class Mirror {
    protected String id;
    protected String frameType;
    protected String frameColor;
    protected double price;

    public Mirror() {};
```

```
    public Mirror(String id, String frameType, String frameColor, double price) {
        this.id = id;
        this.frameType = frameType;
        this.frameColor = frameColor;
        this.price = price;
    }

    public String toString() {
        return "id: " + id + "\n" +
            "Frame Type: " + frameType + "\n" +
            "Frame Color: " + frameColor + "\n" +
            "Price: RM" + price + "\n";
    }
}
```

Part B

```
public class RectangleMirror extends Mirror {
    private int width;
    private int length;

    public RectangleMirror() {};
    public RectangleMirror(String id, String frameType, String frameColor double price, int width, int length) {
        super(id, frameType, frameColor, price);
        this.width = width;
        this.length = length;
    }

    public String toString() {
        return "Rectangle Mirror" + "\n" +
            super() +
            "Width: " + width + "cm\n" +
            "Length: " + length + "cm\n";
    }
}
```

Part C

```
public class MirrorDriver {
    public static void main(String args[]) {
        Mirror[] mirrors = {new RectangleMirror("R1234", "Wood", "White", 480, 52, 162), new RoundMirror("C9876", "Steel", "Dark brown", 92, 40)};
        System.out.println();
        for (Mirror m : mirrors) {
            System.out.println(m);
        }
    }
}
```

Question 4

Part A

Abstract class	Interface
Can have abstract and non-abstract methods	Can only have abstract methods.
Does not support multiple inheritance	Supports multiple inheritance
Can have any variables.	Can only have static and final variables

Part B

```java
public interface Button {
    void onClick();
    void mouseOver();
}
```

```java
public class SurveyForm implements Button {
    @Override
    public void onClick() {
        System.out.print("Thank you for completing our survey!");
    }

    @Override
    public void mouseOver() {
        System.out.println("Click here to submit the survey form.");
    }
}
```

Part C

```java
public static boolean verifyBookId(String id) {
        if (id.length() != 10) {return false;}
        for (int i = 0; i < 5; i++) {
            if (!Character.isDigit(id.charAt(i))) {return false;}
        }
        if (id.charAt(5) != '_') {return false;}
        for (int i = 6; i < 10; i++) {
            if (!Character.isDigit(id.charAt(i))) {return false;}
        }
        int year = Integer.parseInt(id.substring(6,10));
        return year >= 2013 && year <= 2018;
    }
```