CHOO TIAN LONG 24068668

WIX1002 Fundamentals of Programming
Tutorial 9 Inheritance

1. Write statements for each of the following
    a. Write a static method compare that return true if two objects parameter (Student s, Teacher t) are belongs to the same class. Otherwise return false.

```
public static boolean isSameClass(Student s, Teacher t){
   return s.getClass() == t.getClass();
}
```

    b. Write a static method isClass that return true if the object parameter (Student s) is belong to any descendent class of Person. Otherwise return false.

```
public static boolean isClass(Student s){
   return s instanceof Person;
}
```

2. Define a class Organism. The class contains the initial size of the organism and the growth rate. Create a constructor to initialize the instance variables. Then**, d**efine a class Animal. Animal is an organism that has extra instance variable which is the amount of eating need. Create a constructor to initialize the instance variable and a method to display the Animal instance variables.

```
class Organism {
   protected double size, growthRate;

   public Organism(double size, double growthRate){
      this.size = size;
      this.growthRate = growthRate;
   }
}

class Animal extends Organism{
   protected double amountOfEatingNeed;

   public Animal(double size, double growthRate, double amountOfEatingNeed){
      super(size, growthRate);
      this.amountOfEatingNeed = amountOfEatingNeed;
```

```
    }

    public void display(){
        System.out.println("Initial size of the organism: " + this.size);
        System.out.println("Growth rate: " + this.growthRate);
        System.out.println("Amount of Eating Need: " + this.amountOfEatingNeed);
    }
}
```

3. Define a class PaySystem. The class consists of the payrate per hour and the number of hours. The class also contains a method to return the total pay for a given amount of hours and a method to display the total payment. Derive a class RegularPay from PaySystem that has a constructor and did not override any base methods. Derived a class SpecialPay from PaySystem that override the base method and return the total pay plus 30% commission.

```
class PaySystem{
    protected double payRate;
    protected int hour;

    public PaySystem(double payRate, int hour){
        this.payRate = payRate;
        this.hour = hour;
    }

    public double totalPayment(){
        return (payRate*hour);
    }

    public void displayTotalPayment(){
        System.out.printf("Total payment: RM%.2f\n", totalPayment());
    }
}

class RegularPay extends PaySystem{
    public RegularPay(double payRate, int hour){
```

```
        super(payRate, hour);
    }
}

class SpecialPay extends PaySystem{
    public SpecialPay(double payRate, int hour){
        super(payRate, hour);
    }

    @Override
    public double totalPayment(){
        return payRate * hour * 1.3;
    }
}
```

4. Define a class PurchaseSystem. The class consists of product code, unit price, quantity and total price. Besides the class consists of a method to compute the total price and a display method. Derived a class SugarPurchase from PurchaseSystem. This new class add a sugar weight attributed and override the method to compute the total price as unit price*quantity*sugar weight.

```
class PurchaseSystem{
    protected String productCode;
    protected double unitPrice;
    protected int quantity;
    protected double totalPrice = 0;

    public PurchaseSystem(String code, double price, int quantity){
        this.productCode = code;
        this.unitPrice = price;
        this.quantity = quantity;
    }

    public void calculateTotalPrice(){
        totalPrice = unitPrice * quantity;
    }
```

```
    public void displayTotal(){
        System.out.printf("Total price: %.2f\n", totalPrice);
    }
}

class SugarPurchase extends PurchaseSystem{
    protected double sugarWeight;

    public SugarPurchase(String code, double price, int quantity, double sugarWeight){
        super(code, price, quantity);
        this.sugarWeight = sugarWeight;
    }

    @Override
    public void calculateTotalPrice(){
        totalPrice = unitPrice * quantity * sugarWeight;
    }
}
```