WIX1002 Fundamentals of Programming
Tutorial 8 Class

1. Write statements for each of the following
   a. Define a class Student.

```
class Student {

}
```

   b. Declare the instance variable that was used to store the contact number.

```
private String contactNumber;
```

   c. Create the constructor that initializes the contact number to null.

```
public Student(){
   this.contactNumber = null;
}
```

   d. Create another constructor that assign the parameter value to the contact number.

```
public Student(String contactNumber){
   this.contactNumber = contactNumber;
}
```

   e. Create an accessor and mutator method for the contact number.

```
public String getContactNumber(){
   return this.contactNumber;
}

public void setContactNumber(String contactNumber){
   this.contactNumber = contactNumber;
}
```

   f. Create a method that is used to display the contact number.

```
public void printContactNumber(){
   System.out.println("Contact Number : " + this.contactNumber);
}
```

   g. Create an object of the class Student.

```
Student person = new Student();
```

h. Change the contact number using the mutator method.

```
person.setContactNumber("0123456789");
```

i. Create an object of the class Animal.

```
Animal animal = new Animal();
```

j. Create an object of the class Animal that used to represent a cat.

```
Animal cat = new Animal();
```

k. Create an object of the class Number with the value 20 and 40.

```
class Number{
   private int value;

   public Number(){
      this.value = 0;
   }

   public Number(int value){
      this.value = value;
   }
}

Number num1 = Number(20);
Number num1 = Number(40);
```

2. Write statements for each of the following

   a. Define a class Digit.

```
Class Digit{

}
```

   b. Declare the instance variable that used to store a number.

```
private int number;
```

   c. Create a constructor that assign the parameter value to the number.

```
public Digit(int number){
    this.number = number;
}
```

d. Create a digitMultiplication method that returns the multiplication of the number. If the number is 1345, the method will return 60.

```
public int digitMultiplication() {
    int mul = 1; // multiplication
    int tmp = this.number; // temporary

    while(tmp > 0) {
        mul *= (tmp % 10);
        tmp /= 10;
    }

    return mul;
}
```

e. Create a method that is used to display the digit multiplication of the number.

```
public void displayDigitMultiplication(){
    System.out.println(digitMultiplication);
}
```

f. Create a tester class that displays the digit multiplication of 4567.

```
public class q2 {
        public static void main(String[] args) {
                new Digit(4567).displayDigitMultiplication();
        }
}
```

3. Create a class that used to represent the 2 dimension coordinate system. The class consists of constructors, instance variables, accessor and mutator method and an output method that display the x-coordinate and y-coordinate.

```
class Coordinate{
    private int x;
    private int y;

    public Coordinate(){
        this.x = 0;
        this.y = 0;
    }
    public Coordinate(int x, int y){
        this.x = x;
        this.y = y;
    }

    public int getX(){
        return this.x;
    }

    public int getY(){
        return this.y;
    }

    public void setX(int x){
        this.x = x;
    }

    public void setY(int y){
        this.y = y;
    }

    public String toString(){
        return "(" + this.x + ", " + this.y + ")";
    }
}
```

4. Create a class Payment that accept different type of payment methods such as cash payment, cheque payment and credit card payment. For cash payment, the class accepts the amount in cash; for cheque payment, the class accepts the amount and the cheque number; for credit card payment, the class accepts the amount, card holder name, cardType, expiration date and validation code. Use the same method name for the payment.

```
import java.time.LocalDate;

class Payment {
    private double amount;

    public Payment() {
        this.amount = 0;
    }

    public Payment(double amount) {
        this.amount = amount;
    }

    public double getAmount() {
        return this.amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }
}

class Cash extends Payment {
    public Cash() {
        super();
    }

    public Cash(double amount) {
        super(amount);
    }

    public String toString(){
        return "Paid in Cash | Amount : " + this.getAmount();
    }
}

class CreditCard extends Payment {
    private String cardHolderName;
    private String cardType;
    private LocalDate expirationDate;
    private int validationCode;

    public CreditCard() {
        super();
        this.cardHolderName = null;
        this.cardType = null;
        this.expirationDate = null;
        this.validationCode = 0;
    }

    public CreditCard(double amount, String cardHolderName, String cardType, String
expirationDate, int validationCode) {
        super(amount);
        this.cardHolderName = cardHolderName;
        this.cardType = cardType;
        this.expirationDate = LocalDate.parse(expirationDate);
        this.validationCode = validationCode;
    }

    public String getCardHolderName() {
        return this.cardHolderName;
    }

    public void setCardHolderName(String cardHolderName) {
        this.cardHolderName = cardHolderName;
    }

    public String getCardType() {
        return this.cardType;
    }

    public void setCardType(String cardType) {
        this.cardType = cardType;
    }

    public LocalDate getExpirationDate() {
        return this.expirationDate;
    }

    public void setExpirationDate(String expirationDate) {
        this.expirationDate = LocalDate.parse(expirationDate);
    }

    public int getValidationCode() {
        return this.validationCode;
    }

    public void setValidationCode(int validationCode) {
        this.validationCode = validationCode;
    }

    public String toString(){
        return "Paid in Credit Card"
                + "\n" + "Amount : " + this.getAmount()
                + "\n" + "Card Holder Name : " + this.cardHolderName
                + "\n" + "Card Type : " + this.cardType
                + "\n" + "Expiration Date : " + this.expirationDate
                + "\n" + "Validation Code : " + this.validationCode;
    }
}

class Cheque extends Payment {
    private String chequeNumber;

    public Cheque() {
        super();
        this.chequeNumber = null;
    }

    public Cheque(double amount, String chequeNumber) {
        super(amount);
        this.chequeNumber = chequeNumber;
    }
    public String getChequeNumber() {
        return this.chequeNumber;
    }

    public void setChequeNumber(String chequeNumber) {
        this.chequeNumber = chequeNumber;
    }

    public String toString() {
        return "Paid in Cheque"
                + "\n" + "Amount : " + this.getAmount()
                + "\n" + "Cheque Number : " + this.chequeNumber;
    }
}
```

5. Create a class Connection. The Connection class keeps track of the number of connections to the server. Whenever an object is created, a connection is established. The class has a disconnect method and a display method that display the number of connections to the server.

```
class Connection{
    private int numberOfConnection;

    public Connection(){
        this.numberOfConnection = 0;
    }

    public Connection(numberOfConnection){
        this.numberOfConnection = numberOfConnection;
    }

    public void addConnection(){
        this.numberOfConnection++;
    }

    public void addConnection(addConnection){
        this.numberOfConnection += addConnection;
    }

    public void setNumberOfConnection(numberOfConnection){
        this.numberOfConnection = numberOfConnection;
    }

    public int getNumberOfConnection(){
        return this.numberOfConnection;
    }

    public String toString(){
        return "Number of Connection to Server : " + this.numberOfConnection;
    }

}
```