Basic Instructions

- This programming assignment is intended to test your knowledge of Java programming concepts.
- You are required to use Java 8 or higher to complete this assignment. But the recommended version is the latest LTS version Java 17. We recommend installing the AdoptOpenJDK which can be found at https://adoptium.net/.
- You should create a public GitHub repository for this assignment and create a separate folder for each task. Please ensure that your repository is properly named and that each task is properly labelled.
- Your program should be well-designed, well-organized, and easy to understand. You should use proper naming conventions for variables, methods, and classes, and you may include comments where necessary. But the best practice is to write code that makes sense to the reader ;).
- You are encouraged to write unit tests to test the functionality of your program. Please ensure that your unit tests are properly labelled and placed in a separate folder within your relevant task folder.
- You should test your program thoroughly to ensure that it works correctly in all applicable scenarios.
- Your program should be your own original work. You are not allowed to copy code from other sources, including online resources(No ChatGPT please!) and other participants of the training programme.
- At the completion of this assignment a one on one session will be conducted to review your code and provide feedback.
- Happy Coding…Good Luck!

Java Basics - Assignment 01

1. Each of the Java files A,B,C,D,E on this page represents a complete source file. Your job is to play compiler and determine whether each of these files will compile. If they won't compile, how would you fix them?

A.

Java
```java
class Exercise1a {
  public static void main(String[] args) {
    int x = 1;
    while (x < 10) {
      if (x > 3) {
        System.out.println("big x");
      }
    }
  }
}
```

B.

Java
```java
public static void main(String [] args) {
  int x = 5;
  while ( x > 1 ) {
    x = x - 1;
    if ( x < 3) {
      System.out.println("small x");
    }
  }
}
```

C.

```java
class Exercise1c {
  int x = 5;
  while (x > 1) {
    x = x - 1;
    if (x < 3) {
      System.out.println("small x");
    }
  }

}
```

D.

```java
class Books {
  String title;
  String author;
}

class BooksTestDrive {
  public static void main(String[] args) {
    Books[] myBooks = new Books[3];
    int x = 0;
    myBooks[0].title = "The Grapes of Java";
    myBooks[1].title = "The Java Gatsby";
    myBooks[2].title = "The Java Cookbook";
    myBooks[0].author = "bob";
    myBooks[1].author = "sue";
    myBooks[2].author = "ian";
```

```
    while (x < 3) {
      System.out.print(myBooks[x].title);
      System.out.print(" by ");
      System.out.println(myBooks[x].author);
      x = x + 1;
    }
  }
}
```

E.

Java

```
class Hobbits {
  String name;

  public static void main(String[] args) {
    Hobbits[] h = new Hobbits[3];
    int z = 0;

    while (z < 4) {
      z = z + 1;
      h[z] = new Hobbits();
      h[z].name = "bilbo";
      if (z == 1) {
        h[z].name = "frodo";
      }
      if (z == 2) {
        h[z].name = "sam";
      }
      System.out.print(h[z].name + " is a ");
      System.out.println("good Hobbit name");
    }
  }
}
```

2. Assume that you are a software developer working for a company that provides an online banking platform. Your task is to develop a Java program to handle customer accounts.
    - Create a Java class named "BankAccount" that has properties for account number, account holder name, account balance, and account type. Include a constructor that sets the values of these properties, and methods to get and set the values of each property.
    - Create a Java class named "Transaction" that has properties for transaction ID, transaction date, transaction amount, and transaction type. Include a constructor that sets the values of these properties, and methods to get and set the values of each property.
    - Create a Java class named "Customer" that has properties for customer ID, customer name, email address, and phone number. Include a constructor that sets the values of these properties, and methods to get and set the values of each property.
    - Create a Java class named "Bank" that has a list of customers and a list of bank accounts. Include methods to add and remove customers and bank accounts, as well as methods to display customer and bank account information.
    - Write a Java program that allows a user to enter customer information and bank account details, including account type and initial balance. The program should validate the input and create a new bank account for the customer.
    - Add methods to the "BankAccount" class that allow the user to deposit and withdraw money from the account. The program should validate the input and update the account balance accordingly.
    - Add methods to the relevant class that allow the user to view the transaction history of a bank account. The program should display the transaction ID, date, amount, and type for each transaction.
    - Add error handling to your Java program, including validation for incorrect input and handling of exceptions. Test your Java program using different input scenarios, including incorrect input and negative account balances.

3.

    - Create a class named Vehicle with the following attributes:
        - make (String)
        - model (String)
        - year (int)
        - color (String)
        - price (double)

    - Define appropriate constructors and getter/setter methods for each attribute.
- Create a subclass named Car that extends Vehicle and adds the following attributes:
    - numDoors (int)
    - numPassengers (int)
    - isConvertible (boolean)
    - Define appropriate constructors and getter/setter methods for each attribute.
- Create another subclass named Truck that extends Vehicle and adds the following attributes:
    - bedLength (int)
    - payloadCapacity (double)
    - Define appropriate constructors and getter/setter methods for each attribute.
- Create a class named Inventory with the following attributes:
    - vehicles (an array of Vehicle objects)
    - Define appropriate constructors and getter/setter methods for the vehicles attribute.
- In the Inventory class, create a method named getAveragePrice that returns the average price of all vehicles in the inventory.
- In the Inventory class, create a method named searchByMakeAndModel that takes in a make and model (both strings) and returns an array of Vehicle objects that match the make and model.
- Create a main method that creates an inventory object, adds some vehicles to it (both cars and trucks), and tests the methods in the Inventory class.

Your assignment should demonstrate the following object-oriented concepts:

- Inheritance
- Encapsulation
- Polymorphism
- Abstraction

4. Write a program that reads a string from the user and prints the frequency of each character in the string using a selected collection from the Java Collection Framework. You are required to provide justification for your selection of the collection. The end result should be something as follows. For example, if the input string is "hello world", the output should be:
    - h: 1
    - e: 1
    - l: 3
    - o: 2
    - w: 1
    - r: 1
    - d: 1

5. Write a program that reads a list of names from the user and prints them back in the alphabetical order. You need to use an intermediate collection to store user input. Justify your selection and decisions after the implementation.

6. Write a program that simulates a train travelling along a track with multiple stations. Use a suitable collection for the scenario to store the stations and allow the train to start and stop at any station along the way. The program should print all the stations available and accept user input for start and destination stations. Then the program should print stations along the journey from start to finish. The program should terminate at the end station.