Lab Test

Date: 26th June 2023, 10am

Part 1
Instructions

1. The main functions for the questions are provided in the Q1, Q2 and Q3 classes. You are to provide the necessary code to compile the classes successfully. You can edit the classes, but are not encouraged to edit the assertions.
2. The assertions are provided in the main function to help you verify your output. You are encouraged to test your code with test cases of your own. All submitted code will be tested against other test cases during marking.
3. Assertion is disabled by default in Java. To enable it, you need to run your code by -ea or -enableassertions option.
4. Do not edit the class/code/function where it is stated.
5. Copy and paste the "submission code" into the Google submission form provided in the correct text fields.
6. You can make multiple submission before the end of the lab test. Only the latest submission will be recorded and marked.
7. Your submitted Java source code must include all the necessary import statements so that your classes can be compiled into the respective .class files.
8. Failure to follow the submission instructions will be granted ZERO mark.
9. Submission code that failed to be compiled will be granted ZERO mark.
10. Submission after the end of the lab test will be given ZERO mark.

Question 1
Inheritance and Polymorphism (6m)

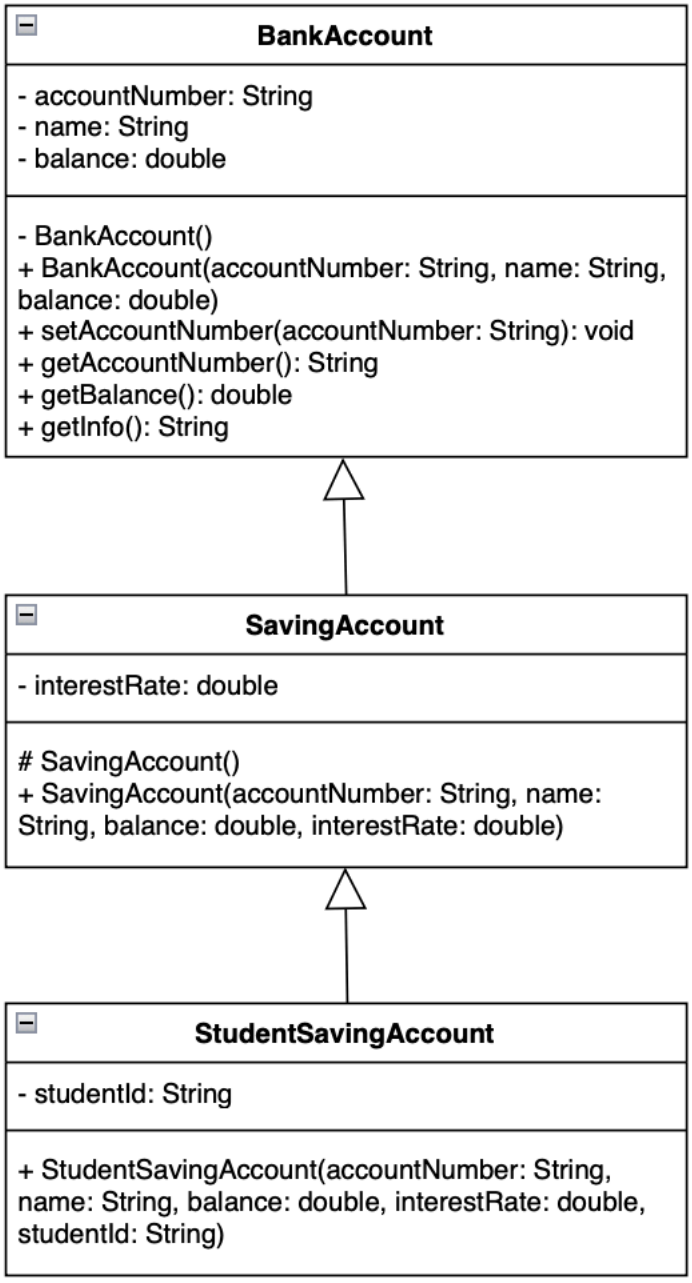

Figure 1 above shows the UML diagram of class BankAccount, SavingAccount and StudentSavingAccount. The BankAccount class is the superclass of SavingAccount, and the SavingAccount is the superclass of StudentSavingAccount.

The StudentSavingAccount class is cater for the student customers and they will get RM100 credit in the balance upon registering an account with the bank.

The Q1 and BankAccount class are provided to you. Do not edit the BankAccount class.

Your task is to provide the SavingAccount and StudentSavingAccount classes and clear all assertions in Q1 class.

Submit your **SavingAccount.java** and **StudentSavingAccount.java** (including import statements if any) to the provided Google form according to the name of the class. Zero mark if your classes failed to be compiled into the respective .class file.

```
// Q1.java
// Inheritance and polymorphism (6m)
public class Q1 {
    public static void main(String[] a){
        int size = 4;
        double interestRate = 0.04;
        BankAccount[] acc = new BankAccount[size];

        acc[0] = new BankAccount("101","Bob B", 1000);
        acc[1] = new SavingAccount("102", "James B", 2000.0, interestRate);
        acc[2] = new StudentSavingAccount("103", "Siti S", 20.0, interestRate ,
"230421-1118");
        for(int i=0; i<size; i++) {
            if(acc[i] != null)
                System.out.println(acc[i].getInfo());
        }
        assert acc[0].getInfo().equals("Account number: 101, name: Bob B, balance:
1000.0"):
                "Failed BankAccount 101, do NOT edit BankAccount.java";
        assert acc[1].getAccountNumber().equals("N-102"): "failed SavingAccount N-102";
        assert acc[2].getAccountNumber().equals("N-S-103"): "failed
StudentSavingAccount N-S-103";
        assert acc[1].getBalance() == 2000:"failed SavingAccount balance";
        assert acc[2].getBalance() == 120:"failed StudentSavingAccount balance";

        acc[1].setAccountNumber("202");
        System.out.println(acc[1].getInfo());
        assert acc[1].getAccountNumber().equals("N-202"): "failed SavingAccount N-202";

        acc[3] = acc[2];
        acc[3].setAccountNumber("203");
        System.out.println(acc[2].getInfo());
        assert acc[2].getAccountNumber().equals("N-S-203"): "failed
StudentSavingAccount N-S-203";

        System.out.println("\nSubmission code: SavingAccount.java and
StudentSavingAccount.java\n");
    }

}
```

```java
//BankAccount.java
public class BankAccount { //Do not edit BankAccount class
    private String accountNumber;
    private String name;
    private double balance;

    private BankAccount() {}

    public BankAccount(String accountNumber, String name, double balance) {
        this.accountNumber = accountNumber;
        this.name = name;
        this.balance = balance;
    }

    protected void setAccountNumber(String accountNumber) {
        this.accountNumber = accountNumber;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public String getInfo() {
        return "Account number: " + accountNumber + ", name: " + name +
            ", balance: " + balance;
    }

}
```

Question 2
Interface and making a copy of object class (4m)

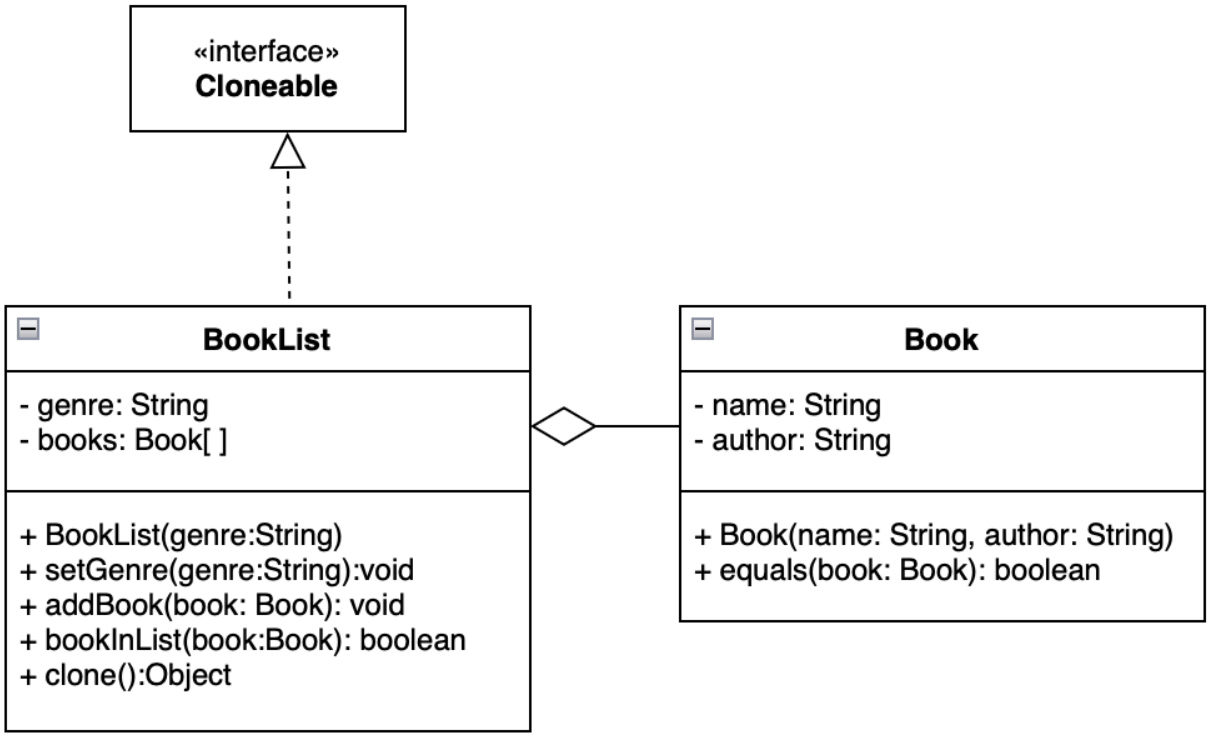

Figure 2 above shows the UML diagram of class BookList and Book. The Book class is provided to you. Your task is to write the BookList class to clear all assertions in Q2 class.

Class BookList contains a list of books (no limit to the number of books in the list) and implements Cloneable interface. You are free to use any collections or data structure from Java API, and do not forget to include the import statement in your submission code.

The addBook function is used to add a Book object into the list.

The bookInList function is used to check if a book object can be found in the list; the function return true if the book can be found in the list, false otherwise.

The clone function is used to make a deep copy of the BookList object.

Submit your BookList.java (including import statements if any) to the Google form. Zero mark if your class failed to be compiled into the .class file.

```java
//Q2.java
//Interface and making a copy of Object class (4m)
public class Q2 {
    public static void main(String[] a) {
        Book book1 = new Book("Jane Austen", "Pride and Prejudice");
        Book book2 = new Book("George Orwell","Nineteen Eighty-Four");

        BookList blist1 = new BookList("Fiction");
        blist1.addBook(book1);
        System.out.println(blist1);
        assert blist1.toString().contains("Fiction:(author: Jane Austen, title: Pride
and Prejudice)"): "Failed BookList 1 output";

        //Making a deep copy of blist1
        BookList blist2 = (BookList)blist1.clone();
        blist2.setGenre("Classics");
        blist2.addBook(book2);
        //No changes in blist1 book list
        System.out.println(blist1);
        assert blist1.bookInList(book1): "Failed BookList1 book1 (after clone) output";
        assert !blist1.bookInList(book2): "Failed BookList1 book2 (after clone)
output";
        // blist2 has 2 books in the list
        System.out.println(blist2);
        assert blist2.bookInList(book1): "Failed BookList2 book1 output";
        assert blist2.bookInList(book2): "Failed BookList2 book2 output";

        System.out.println("\nSubmission code: BookList.java \n");
    }
}
```

```java
//Book.java
public class Book { //Do not edit Book class
    private String author;
    private String title;
    public Book(String author, String title){
        this.author = author;
        this.title = title;
    }
    @Override
    public boolean equals(Object o) {
        if( o == this ) return true;
        if (!(o instanceof Book)) return false;
        Book b = (Book) o;
        if(this.toString().equals(b.toString())) return true;
        return false;
    }
    @Override
    public String toString() {
        return "(author: " + author + ", title: " + title +")";
    }
}
```

Question 3
Exception handling(5m)

Exception handling

You are given class Q3 with main function. You task is to provide the ArrayCalculator class to clear all the assertions in the main function. The ArrayCalculator containing a double array and a getAverage function. The getAverage function calculates the average value of the double array up to a specific size, and return the value in String data type in 2 decimal places. You must use exception handling to handle the cases where the array are accessed out-of-bound.

Submit your **ArrayCalculator.java** (including import statements if any) to the Google form.

```
// Q3.java
// Exception handling (5 marks)
public class Q3 {

    public static void main(String[] a){
        double arr[] = {3.3, 9.2, 5, 26.4};
        ArrayCalculator cal = new ArrayCalculator(arr, arr.length);
        for(int i=0; i<5; i++)
            System.out.println(cal.getAverage(i));

        //Assertions for Q3, do not edit
        assert cal.getAverage(0).equals("Error! Cannot divide by zero."): "Failed 0";
        assert cal.getAverage(1).equals("3.30"): "Failed 1";
        assert cal.getAverage(2).equals("6.25"): "Failed 2";
        assert cal.getAverage(3).equals("5.83"): "Failed 3";
        assert cal.getAverage(4).equals("10.98"): "Failed 4";
        assert cal.getAverage(5).equals("Error! Element not accessible in the array."):
"Failed 5";
        assert cal.getAverage(10).equals("Error! Element not accessible in the
array."): "Failed 10";

        System.out.println("Submission code: ArrayCalculator.java");

    }
}
```