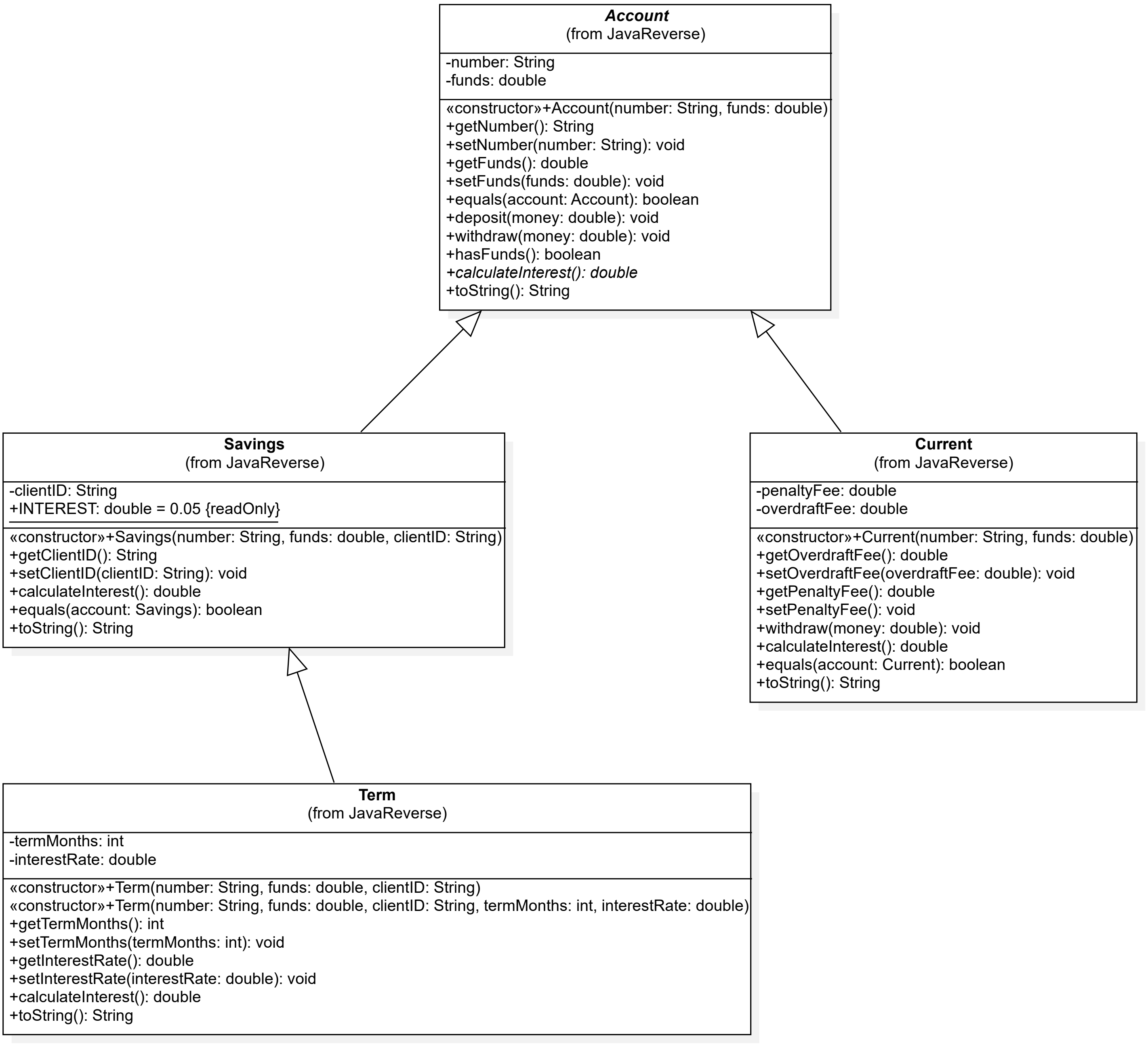
Name [ ] Date [ ]

## Instructions

The **`Account`** and **`TestAll`** classes are given to you. Write the ***`Current`***, ***`Savings`*** and ***`Term`*** classes according to the class diagram below:


Account
(from JavaReverse)
-number: String
-funds: double
«constructor»+Account(number: String, funds: double)
+getNumber(): String
+setNumber(number: String): void
+getFunds(): double
+setFunds(funds: double): void
+equals(account: Account): boolean
+deposit(money: double): void
+withdraw(money: double): void
+hasFunds(): boolean
+calculateInterest(): double
+toString(): String
Savings
(from JavaReverse)
-clientID: String
+INTEREST: double = 0.05 {readOnly}
«constructor»+Savings(number: String, funds: double, clientID: String)
+getClientID(): String
+setClientID(clientID: String): void
+calculateInterest(): double
+equals(account: Savings): boolean
+toString(): String
Current
(from JavaReverse)
-penaltyFee: double
-overdraftFee: double
«constructor»+Current(number: String, funds: double)
+getOverdraftFee(): double
+setOverdraftFee(overdraftFee: double): void
+getPenaltyFee(): double
+setPenaltyFee(): void
+withdraw(money: double): void
+calculateInterest(): double
+equals(account: Current): boolean
+toString(): String
Term
(from JavaReverse)
-termMonths: int
-interestRate: double
«constructor»+Term(number: String, funds: double, clientID: String)
«constructor»+Term(number: String, funds: double, clientID: String, termMonths: int, interestRate: double)
+getTermMonths(): int
+setTermMonths(termMonths: int): void
+getInterestRate(): double
+setInterestRate(interestRate: double): void
+calculateInterest(): double
+toString(): String

Work through the test from the beginning. Your Time class should build and grow –do not start a new program for each point.

During this test, you may use any resources that you have created, but you may **not** use Internet.

Refer to the UML class diagram on this page when writing your code.

<<< Please Turn Over >>>

## Marks Distribution

1. Successfully implement ***inheritance*** in all your classes as shown in the class diagram.
2. Successfully implement ***all fields/variables*** in all your classes
3. Successfully use ***data hiding*** as required by the UML class diagram given.
4. Successfully implement ***constructors*** in all your classes as per the class diagram.
5. Use one constructor in the *Term* class to initialise the term as 6 months by default, and the interest rate to 2.7%, and overload the constructor with another one allowing for the customisation/initialisation of the term and interest rate.
6. Successfully implement ***accessors*** as shown in the class diagram.
7. Successfully implement ***mutators*** as required by the class diagram.
8. Successfully implement ***toString*** methods in all your classes. They should return Strings as similar as possible to the display output on next page/expected output in the program.
9. Successfully implement methods to calculate ***interest rates*** in all your classes:
   In the *Current* class, *calculateInterest* always returns zero;
   In the *Savings* class, use *funds * INTEREST* (0.05) to calculate the interest
   In the *Term* class, the interest rate is calculated with the following formula:
   (funds * interestRate * termMonths) / 1200
10. Successfully implement ***equals*** methods in the relevant classes as shown in the class diagram.
    In the *Current* class, both account numbers must be the same...
    In the *Savings* and *Term* classes, the account number and client IDs must be the same…
    ...for the objects to be considered equal

<<< Please Turn Over >>>

## Display Output

```
Expected output: ClientID: pepeperez  Acct: Savings  Number: SA-1111  Funds: $5,000.00
>>> Your output: ClientID: pepeperez  Acct: Savings  Number: SA-1111  Funds: $5,000.00

Expected output: Withdrawing $5k -> Funds: false -> Deposit $2kFunds -> Funds? true
>>> Your output: Withdrawing $5k -> Funds: false -> Deposit $2kFunds -> Funds? true

Expected output: Savings acct. interest rate(%) = 0.05 -> Interest for SA-1111 = $100.0
>>> Your output: Savings acct. interest rate(%) = 0.05 -> Interest for SA-1111 = $100.0

Testing Savings account class equals method
Expected output: mySavingsAcc.equals(dodgy) = true; mySavingsAcc.equals(yourSavings) = false
>>> Your output: mySavingsAcc.equals(dodgy) = true; mySavingsAcc.equals(yourSavings) = false

Expected output: Acct: Current  Number: CA-1234  Funds: $3,500.00  Overdraft penalty fee: $0.00
>>> Your output: Acct: Current  Number: CA-1234  Funds: $3,500.00  Overdraft penalty fee: $0.00

Overdraft fee: $300.0  Expected output: $300.0
Overdraft fee: $875.0  Expected output: $875.0

Expected output: Withdrawing $5k -> Funds: false -> Overdraft penalty fee: $2,375.00
>>> Your output: Withdrawing $5k -> Funds: false -> Overdraft penalty fee: $2,375.00
Testing Current account class equals method
Expected output: dos.equals(tres) = true; myCurrentAcc.equals(dos) = false
>>> Your output: mySavingsAcc.equals(dodgy) = true; myCurrentAcc.equals(dos) = false

Expected output:
ClientID: ricardo_rojo  Acct: Term  Number: TA-9876  Funds: $25,000.00  Term: 6 months  Interest: 2.70%
>>>>Your output:
ClientID: ricardo_rojo  Acct: Term  Number: TA-9876  Funds: $25,000.00  Term: 6 months  Interest: 2.70%
Expected output: Calculated interest earned -> $337.50
>>> Your output: Calculated interest earned -> $337.50

Expected output: Extending to 12 months and doubling interest rate -> 12 months @ 5.40
>>> Your output: Extending to 12 months and doubling interest rate -> 12 months @ 5.40
Expected output: Calculated interest earned -> $1,350.00
>>> Your output: Calculated interest earned -> $1,350.00

Testing Term account class equals method
Expected output: fixed.equals(t3) = true; fixed.equals(t3) = false
>>> Your output: fixed.equals(t3) = true; fixed.equals(t3) = false
```