CS2030S

AY23/24 S2 **Problem Set 0** January 2024

1. In this question, your task is to create an abstraction for a single-digit ternary number, that can only store the values 0, 1, or 2.

   (a) Write a class called `Ternary` with an `int` field named `value`. The field should not be accessible from outside the class. The class should have a constructor that initializes `value` to 0, and a `toString` method that returns the `value` as a `String`.

   Example of how the class can be used:

   ```
   jshell> Ternary t = new Ternary();
   t ==> 0
   ```

   Note: You can use the static method `String::valueOf` to convert an `int` to a `String`. See the Java API for `String` for more information.

   (b) Add a method called `incr` to the class. `incr` should increment `value` by one but wraps around to 0 when the value exceeds 2. The method should not return anything.

   Example of how the class can be used:

   ```
   jshell> Ternary t = new Ternary();
   t ==> 0

   jshell> t.incr()
   jshell> t
   t ==> 1

   jshell> t.incr()
   jshell> t
   t ==> 2

   jshell> t.incr()
   jshell> t
   t ==> 0
   ```

   Comments:

   ```
   class Ternary {
     private int value;

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

     public void incr() {
       this.value = (this.value + 1) % 3;
     }

     @Override
     public String toString() {
       return String.valueOf(this.value);
     }
   }
   ```

2. *This question is adapted from the CS2030S midterm test of AY 21/22 Sem 2.*

   Consider the following Java program:

```
class BankAccount {
  double balance;

  BankAccount(double initBalance) {
    this.balance = initBalance;
  }
}

class Customer {
  BankAccount account;

  Customer() {
    this.account = new BankAccount(0);
  }

  public void deposit(double amount) {
    this.account.balance += amount;
  }

  public boolean withdraw(double amount) {
    if (this.account.balance >= amount) {
      this.account.balance -= amount;
      return true;
    }
    return false;
  }
}
```

   (a) Does this program follow the principle of information hiding? Explain.

   Comments:

   No. The balance information in `BankAccount` is publically accessible. So is the `account` information of a `Customer`.

   (b) Does this program follow the principle of "Tell, Don't Ask?" Explain.

   Comments:

   No. `Customer` directly checks the balance of `BankAccount` and modifies the value. It is asking for the balance from the account and then updates it, rather than telling the account to update its own balance.

   (c) If you think the program violates any of the principles in Parts (a) and (b), revise the program so that it adheres to the principles.

   Comments:

```
class BankAccount {
  private double balance; // make this private

  BankAccount(double initBalance) {
    this.balance = initBalance;
  }
```

```
  public void deposit(double amount) {
    this.balance += amount;
  }

  public boolean withdraw(double amount) {
    if (this.balance >= amount) {
      this.balance -= amount;
      return true;
    }
    return false;
  }
}

class Customer {
  private BankAccount account; // make this private

  Customer() {
    this.account = new BankAccount(0);
  }

  public void deposit(double amount) {
    this.account.deposit(amount); // tell account to do it
  }

  public boolean withdraw(double amount) {
    return this.account.withdraw(amount); // tell account to do it
  }
}
```