Object-Based Programming (25 marks):
Answer both parts below.

The API for the `Money` class is given below. Each instance of this class represents an amount of dollars and cents. The amount of cents will be an integer between 0 and 99 (inclusive), and the amount of dollars will never be less than zero.

Constructor Summary	
`Money()` Constructs a new Money amount with 0 dollars and 0 cents.	
Method Summary	
`boolean`	`equals(Money amount)` Returns `true` if this Money amount is equal to `amount`.
`boolean`	`isGreaterOrEquals(Money amount)` Returns `true` if this Money amount is greater than of equal to `amount`.

The API for the `BankAccount` class is given below. Each instance of this class has a balance with a certain amount of Money.

Constructor Summary	
`BankAccount()` Constructs a new BankAccount with a zero balance (i.e. no Money).	
`BankAccount(Money amount)` Constructs a new BankAccount where the initial balance is equal to the given `amount` of Money.	
Method Summary	
`void`	`deposit(Money amount)` Adds the given `amount` of Money to the current balance.
`Money`	`getBalance()` Returns the amount of Money currently in this BankAccount.
`boolean`	`withdraw(Money amount)` Attemps to subtract the given `amount` of Money from the current balance. Subtracts the `amount` and returns `true` if the current balance is larger than the `amount` to withdraw. Otherwise, returns `false` and charges a penalty of 10 dollars.

The API for the `CashCard` class is given below. Each instance of this class has a balance with a certain amount of Money that must be between zero and the maximum balance allowed on a CashCard (inclusive).

Field Summary	
`static Money`	`MAX_BALANCE` The maximum amount of Money that can be stored on a CashCard.
Constructor Summary	
`CashCard()` Constructs a new CashCard with a zero balance (i.e. no Money).	
`CashCard(Money amount)` Constructs a new CashCard. If the given `amount` of Money is less than the maximum balance, then the balance is set to the `amount`; otherwise, it is set to zero.	
Method Summary	
`boolean`	`add(Money amount)` Attemps to add the given `amount` of Money to the current balance. Adds the `amount` and returns `true` if the available limit is larger than the `amount` to add. Otherwise, returns `false` and sets the current balance to the maximum balance.
`boolean`	`deduct(Money amount)` Attemps to subtract the given `amount` of Money from the current balance. Subtracts the `amount` and returns `true` if the current balance is larger than the `amount` to subtract. Otherwise, returns `false` and sets the current balance to zero.
`Money`	`deductAll()` Returns the amount of Money currently on this CashCard and sets the balance to zero.
`Money`	`getBalance()` Returns the amount of Money currently on this CashCard.
`Money`	`getLimit()` Returns the amount of Money that can be added to this CashCard without exceeding the maximum balance.

Surname:______________ First name:_______________ Student #: ________________

Part 1 (**10 marks**):

The Kingdom of Queensland has decided that it will no longer issue paper money and coins. Instead, it will use CashCards. In order for CashCards to be usable by the public, it must be possible to transfer Money between CashCards.

Write a code fragment in JAVA that will determine if the given amount of Money can be and is transferred between CashCards. Your code must set the variable `successful` to `true` after a successful transfer and `false` otherwise. To be a successful transfer, the `amount` of Money being transferred cannot be greater than the balance of the transferring CashCard (`from`) or cause the receiving CashCard (`to`) to exceed the maximum allowed balance. No changes to the CashCards should occur on an unsuccessful transfer attempt.

```
// transfer
boolean successful;
CashCard from;
CashCard to;
Money amount;
```

Surname:______________ First name:_______________ Student #: ________________

Part 2 (**15 marks**):

It will also be necessary for the public to be able to buy CashCards at their bank.

Write a code fragment in JAVA that will assign to `newCard` a new CashCard with a balance equal to the given `amount` of Money that will be withdrawn from the given BankAccount (`account`). If the requested `amount` is greater than the maximum balance allowed on a CashCard or the current balance of the given `account`, then `newCard` should be set to `null` and the current balance of the `account` should be left unchanged.

```
// buyCard
CashCard newCard;
Money amount;
BankAccount account;
```