<u>Answers</u>

1. **Run your Console application, what is it doing? What is being outputted to the console?**

   There I s 2 accounts one is with good security features (SecureBankAccount) and other one is with poor security feature(DodgyBankAccount).

   In this console application we try to

   - Check the current balance.
   - Deposit an amount of 30.
   - Check current balance again.
   - During each deposit of 30, 50 is being rewarded.
   - Both of the account Users are trying to add reward without deposit (trying to penetrate the security).
   - User is trying to change the account balance by accessing the variable to add lots of money directly(fraud activity).

   <u>Console Output:</u>

```
--- DODGY BANK ACCOUNT ---

Mmm...I spot a dodgy bank account! Let's make one!
What is my current balance?
Your account balance is 0
Let's deposit an amount of 30
Your account balance is 80
Wow! Looks like we get a reward of 50 when we deposit an amount
Let's be naughty and add rewards without depositing!
Your account balance is 230
Wow! We're rich!
Let's make a lot of money right now! Let's change the account balance
directly!
Your account balance is 1000000
?? Weeeeee!!!!

--- SECURE BANK ACCOUNT ---

Mmm...let's make a secure bank account!
What is my current balance?
Your account balance is 0
Let's deposit an amount of 30
Your account balance is 80
Wow! Looks like we get a reward of 50 when we deposit an amount
Let's be naughty and add rewards without depositing!
Oh no :( It looks like we can't do this - it's too secure!
Your account balance is 80
Let's make a lot of money right now! Let's change the account balance
directly!
Oh no :( It looks like we can't do this - it's too secure!
Your account balance is 80
?? Well I guess that's secure!
```

2. **Look at the DodgyBankAccount, this class is not well-encapsulated. Can you note down the problems with how the class is designed, and the ways it is being misused?**

- Here all variables are declared as public variables including account balance and reward, so anyone can easily amend those values.

```
public string AccountNumber;
public int AccountBalance;
public int RewardAmount = 50;
```

- AddReward() method is defined as public method in DodgyBankAccount so user can add rewards without depositing money through execute this method again and again.

```
public void AddReward()
          {
              AccountBalance += RewardAmount;
          }
```

3. **Compare and contrast the DodgyBankAccount and the SecureBankAccount, how is the SecureBankAccount different to the DodgyBankAccount? How is it designed to prevent it from being misused? Are there instances of better method names for clearer abstraction?**

SecureBankAccount is well encapsulated, all of the sensitive data variables are declared with private access modifier, at the same time in the case of **DodgyBankAccount** all sensitive data variables are declared as public. Because of this user can misuse **DodgyBankAccount.**

Table shows the differences between theses 2 classes.

	DodgyBankAccount	SecureBankAccount
Variables	`public string AccountNumber;` `public int AccountBalance;` `public int RewardAmount = 50;`	`private readonly string _accountNumber;` `private const int REWARD_AMOUNT = 50;` `private int _accountBalance;`
AddReward()	`public void AddReward()` `{` `AccountBalance += RewardAmount;` `}`	`private void AddReward()` `{` `_accountBalance += REWARD_AMOUNT;` `}`
Security	Poor Security	Secured