CoBloX hiring challenge

Programming task

Thank you very much for taking on the hiring challenge for the developer position at CoBloX! Our daily work touches quite a few different areas. This exercise is designed to mimic some of the challenges we face in every day work.

How does this work

The challenge consists of two parts:

1. A programming task
2. A research task

This document describes part 1, the programming task of this challenge.

Handing in

Once you are **done** with the **programming task**, please submit it to us by creating a **private** GitHub repository and adding the following people as contributors:

- `https://github.com/bonomat`
- `https://github.com/D4nte`
- `https://github.com/thomaseizinger`

Please include your full name in the README. We will snapshot the repository after your invite. Further changes to the repository after your submission will remain unnoticed by us. We aim to look at your solution within two business days but please do not worry if it takes a little longer.

Questions

If you have any questions at any point, feel free to reach out to the recruiter you are in contact with.

What you should do

Your task is to re-design a simple guessing game so that it *could* be played over the Internet. The game is called "king-or-queen". It's played between two parties who we'll call Alice and Bob and these are the rules:

1. Alice has two cards: a 'King' and a 'Queen'.
2. She starts by secretly choosing one and putting it upside-down on the table in front of her.
3. Bob tries to guess Alice's move by shouting out either 'King' or 'Queen'.
4. Alice reveals her card.
5. If Bob guessed correctly, he wins, otherwise Alice is the winner.

So how could we play this over the Internet? Well, here's the output of a simple program that demonstrates a *naive* way of playing it: (where `Alice => Bob` means that Alice sends this message to Bob over the Internet)

Naive King-or-Queen

```
$ ./king-or-queen
Alice => Bob: I've chosen my card
Bob => Alice: I guess it is King
Alice => Bob: My card is Queen

Alice wins!
```

But there's a problem! The reason this is naive is because it only works if Alice is an honest person. If Alice actually chose 'King' at the start, she could have easily cheated by changing her move to 'Queen' after seeing Bob guessed 'King'.
Your task is to create a **fair** version of this demonstration where we don't have to trust anyone not to cheat. This is where cryptography joins the party: you'll have to find a way to enforce that Alice can't change her card after she "puts it down". Or more specifically, anyone looking at the output of the program should be able to tell that Alice changed her move and declare Bob the winner instead. To do this, you'll have to change the messages Alice and Bob send each other so they have some cryptographic information.

Requirements

1. Your software should be split into a least two parts:
   a) A **library** component that encapsulates the game in a reusable way. Typical to a library, it should:
      - have little opinion on how it is used
      - provide reasonable configuration options
      - allow for customization of certain behaviour
      - etc…

   b) An **executable** component that uses the library component to provide a commandline application for playing the game
2. The executable component should **not** use any network communication.
3. The executable component should take the following data as input:
   a) Alice's move
   b) Bob's move
   c) What Alice reveals as her move

   You can decide on the form of input. (environment variables, STDIN, configuration file, commandline arguments, etc…)
4. The program should print the **full contents of every message Alice and Bob send to each other** as well as indicate who sent the message.
5. At the end, the program should print out who won.
6. Given that Alice's move and what she reveals is not necessarily the same (they are different inputs according to requirement 3), it is possible that Alice tries to cheat on Bob by revealing a different card than what she initially chose as her move. Bob MUST be able to detect that Alice tried to cheat. The program should print out the true winner in this scenario.
7. Bob MUST not be able to cheat either.
8. Choose any programming language you think is a good fit for this problem.

Notes

The programming task is purposely designed to be small. As a result, it is considered a minimum requirement that the solution is correct. But please show off your software engineer skills!

Have fun!