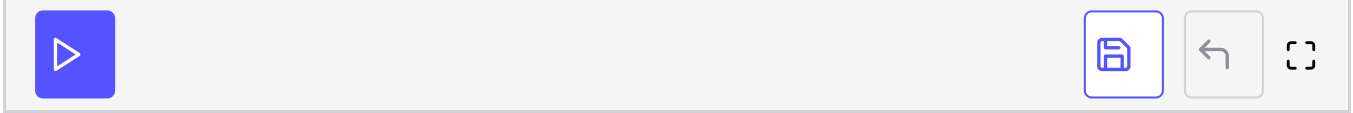
Solution Review: Buzz Game

Let's quickly go over the solution to the Buzz Game Exercise.

```
package main

import (
  "fmt"
  "time"
  "math/rand"
)

func main() {
  channel1 := make(chan string)
  channel2 := make(chan string)

  go func() {
    rand.Seed(time.Now().UnixNano())
    time.Sleep(time.Duration(rand.Intn(500)+500) * time.Millisecond)
    channel1 <- "Player 1 Buzzed"
  }()

  go func() {
     rand.Seed(time.Now().UnixNano())
     time.Sleep(time.Duration(rand.Intn(500)+500) * time.Millisecond)
     channel2 <- "Player 2 Buzzed"
  }()

  select{
    case message1 := <-channel1:
      fmt.Println(message1)
    case message2 := <-channel2:
      fmt.Println(message2)
  }

}
```



Buzz Game

You can see that we unblocked the channel receiving operations by using a `select` statement (**lines 25-30**). Yes, it was that simple!

```
select{
    case message1 := <-channel1:
```

```
        fmt.Println(message1)
    case message2 := <-channel2:
        fmt.Println(message2)
    }
```

The `select` statement chooses operations which are ready from the other end. This solution eliminates the blocking of receiving operations by channels. Hence, whichever channel will be the first to send the `Buzz` signal, the `select` statement will execute the corresponding case.

I hope you are able to appreciate the simplicity of the features in Golang when it comes to solving problems.

In the next lesson, we'll study `WaitGroups` from the `sync` Package.