Swift Advance/ Error handling

Exercise 6

Abhishek Maurya

1. **What is Error Protocol. Create a custom error conforming to error protocol.**
    - Error Protocol is just a type for representing error values that can be thrown.

```
enum UserDetailError: Error {
    case noValidName
    case noValidAge
}
```

2. **Write a failable initialiser for a class which throws a error "Object not able to initialise" description a initialisationFailed case, Catch the error and print its error description.**
    -

```
enum ErrorHandler : Error {
    case initilisationFailed
}
class ErrorCheck {
    var numberInt : Int?
    init? (notZero : Int) {
        if notZero == 0 {
            return nil
        }
        else {
            self.numberInt = notZero
        }
    }
}
var obj = ErrorCheck(notZero: 0)
do{
    if (obj == nil) {
        throw ErrorHandler.initilisationFailed
    }
    else {
        print(obj!.numberInt!)
    }
}
catch {
    print("Object not able to initialise")
}
```

3. **Explain the difference try, try?, try! , make sure to write a program to explain the difference.**
    - try is used to try for the error in the following code
    - try! is used when the user is sure that the following code will surely not throw the error, but if the error thrown then the playground will crash. This can be used without do catch block.
    - try? is used when the user is not sure that it may or may not throw the error , This can be used without do catch block.
    -

4. **Write a program which loads the data from a datasource of 10 employees looks like below, Program would help to give salary bonus to employees. Which is based on some conditions but if employee is not able to satisfy the condition program should throw the error with specific error condition and its description should be printed.**

```
struct toTheNew {
    var empID : Int
    var empName : String
    var empEmail : String
    var joiningDate : Int
    var isPresent : Bool
    var competency : String
    var attendancePercent : Int
}
enum noBonusReasons : Error {
    case EmployeenotPresent(String)
    case Competency(String)
    case notCompletedoneYear(String)
    case notaemployee(String)
    case attendanceLessThenEighty(String)
}
class BonusProgram {
    var TTNEmployees : [toTheNew] = [toTheNew(empID: 101, empName:
"Muskan", empEmail: "muskaan@tothenew.com",
joiningDate: 2018 , isPresent: false, competency: "ios" ,
attendancePercent: 89) ,
                                     toTheNew(empID: 102, empName:
"Mithesh", empEmail: "mithlesh@tothenew.com", joiningDate: 2017 ,
isPresent: true, competency: "Bussiness intelligence" , attendancePercent:
89) ,
                                     toTheNew(empID: 103, empName:
"Ankit", empEmail: "ankitnigam@tothenew.com", joiningDate: 2018 ,
isPresent: true, competency: "ios" , attendancePercent: 90) ,
                                     toTheNew(empID: 104, empName:
"Sachin", empEmail: "sachin@tothenew.com", joiningDate: 2019 , isPresent:
true, competency: "ios" , attendancePercent: 81) ,
                                     toTheNew(empID: 101, empName:
"Muskan", empEmail: "muskaan@tothenew.com", joiningDate: 2018 , isPresent:
false, competency: "ios" , attendancePercent: 89) ,
                                     toTheNew(empID: 105, empName:
"Merry", empEmail: "merry@tothenew.com", joiningDate: 2015 , isPresent:
true, competency: "ios" , attendancePercent: 95)]
    func allowedForBonus (Email :String) throws {
        let PresentYear = 2019
        var eligiblity = 0
        for emp1 in TTNEmployees {
            if(Email == emp1.empEmail) {
                if !emp1.isPresent {
                    eligiblity = 0
```

```
                    throw noBonusReasons.EmployeenotPresent(" is absent")
                }
                else {
                    eligiblity = 1
                }
                if (emp1.competency == "ios" || emp1.competency ==
"android" || emp1.competency == "BigData" || emp1.competency == "AI") {
                    eligiblity = 1
                }
                else {
                    eligiblity = 0
                    throw noBonusReasons.Competency("competency does not
fall under bonus program.")
                }
                if( (PresentYear - emp1.joiningDate) > 0) {
                    eligiblity = 1
                }
                else {
                    eligiblity = 0
                    throw noBonusReasons.notCompletedoneYear("and still to
complete a year with us")
                }
                if (emp1.attendancePercent >= 80) {
                    eligiblity = 1
                }
                else {
                    eligiblity = 0
                    throw noBonusReasons.attendanceLessThenEighty("do not
have attandance more then eighty percent")
                }
                if(eligiblity == 1) {
                    print(emp1.empName ," is eligible for bonus.")
                }
            }
        }
    }
}
var obj1 = BonusProgram()
do {
    try obj1.allowedForBonus(Email: "muskaan@tothenew.com")
}
catch {
    print(error)
}
do {
    try obj1.allowedForBonus(Email: "mithlesh@tothenew.com")
}
catch {
    print(error)
}
```

```
do {
    try obj1.allowedForBonus(Email: "ankitnigam@tothenew.com")
}
catch {
    print(error)
}

do {
    try obj1.allowedForBonus(Email: "sachin@tothenew.com")
}
catch {
    print(error)
}

do{
    try obj1.allowedForBonus(Email: "merry@tothenew.com")
}
catch {
    print(error)
}
```