НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ» ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

ЛАБОРАТОРНА РОБОТА № 7
ВИКОРІСТАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПІДХОДУ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Виконала:
студент групи ФІ-12
Бекешева Анастасія

Contents

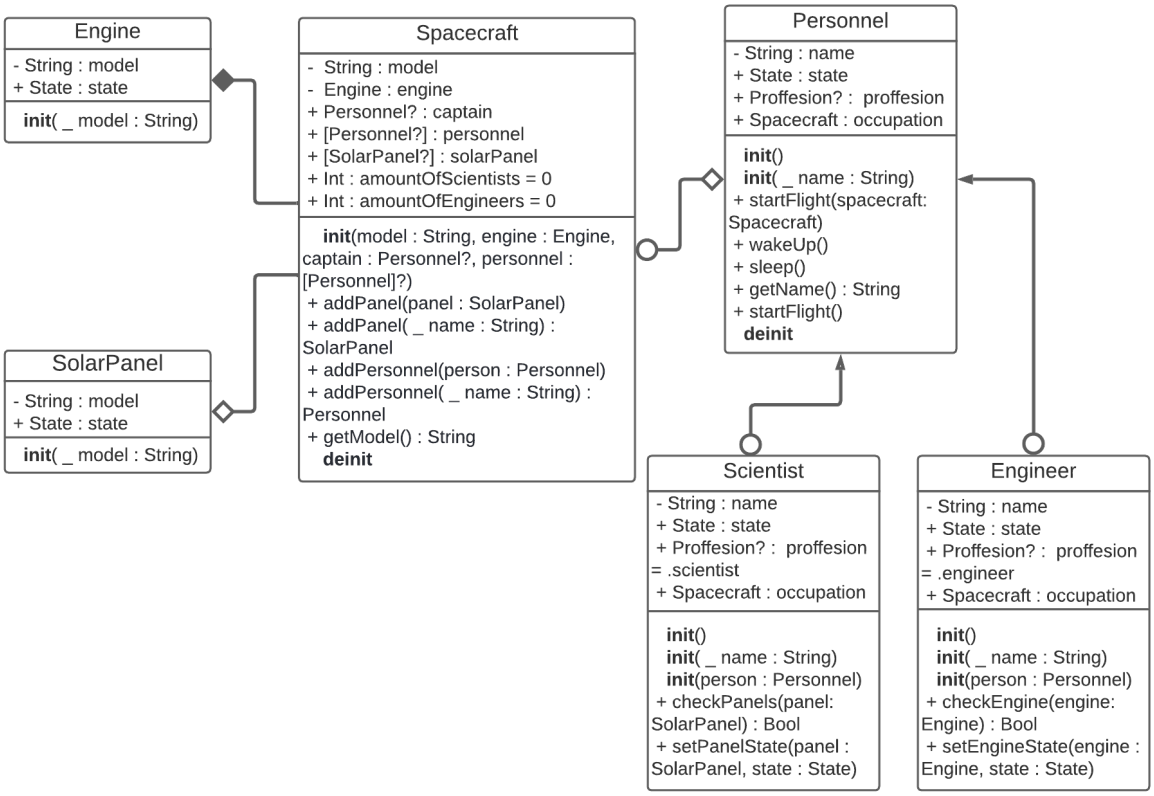
1 Вступ

Дана робота має задачу змоделювати космічний апарат. Її реалізація полягає в моделюванні структури космічного апарта, тобто з чого він скаладється та як працює. Проаналізувавши поставлену задачу, необхідно забезпечити в структурі космічного апарату:

1. Двигун
2. Додаткові елменти, наприклад, сонячні панелі
3. Головного за космічний апарат
4. Персонал, розподілений за професіями

У даній роботі вище перераховані зауваження реалізовані введенням шести класів та встановленням певних відношень між ними.

2 UML-діаграма



3 Код прогарми

3.1 Main.swift

```
//
//  main.swift
//  Spacecraft
//
//  Created by Nastya Bekesheva on 08.04.2022.
//

import Foundation

var spacecrafts: [Spacecraft] = []
var solarpanels: [SolarPanel] = []
var people: [Personnel] = []
var state = true
print("What would you like to do?\n 1. Create Spacecraft\n 2. Create Solar Panel\n 3. Cr
while state{
    let answer = readLine()
    if let answer = Int(answer ?? "w") {
        switch answer{
        case 1:
            print("Set model for Spacecraft: ")
            let model = readLine()
            print("Set model for Spacecraft's Engine: ")
            let engineModel = readLine()
            print("Set name for captain")
            let captain = readLine()

            spacecrafts.append(Spacecraft(model: model ?? "no name", engine: engineModel
            print("Spacecraft created")
        case 2:
            print("Set model for solar panel: ")
            let model = readLine()
            solarpanels.append(SolarPanel(model ?? "no name"))
            print("Success!")
        case 3:
            print("Set name for person: ")
            let name = readLine()
            people.append(Personnel(name ?? "no name"))
            print("Success!")
        case 4:
            print("Set name for person: ")
            let name = readLine()
            people.append(Scientist(name ?? "no name"))
            print("Success!")
        case 5:
            print("Set name for person: ")
            let name = readLine()
            people.append(Engineer(name ?? "no name"))
```

```
    print("Success!")
case 6:
    var s = 0
    var p = 0
    print("Enter model of Spacecraft: ")
    let model = readLine()

    for spacecraft in spacecrafts{
        if spacecraft.getModel() == model{
            s += 1
            print("Enter name of the person: ")
            let name = readLine()
            for person in people {
                if person.getName() == name{
                    p += 1
                    if spacecraft.getModel() != person.occupation?.getModel(){
                        spacecraft.addPersonnel(person: person)
                        print("Successfully added \(name!) to \(model!)!")
                    }
                    else{
                        print("Already on board!")
                    }
                }
            }
            if p == 0{
                print("Failed to find a person with name \(name!)...\nWould you
                let ans = readLine()?.lowercased()
                if ans == "y"{
                    people.append(spacecraft.addPersonnel(name!))
                    print("Success!")
                }
                else if ans == "n"{
                    print("As you wish")
                }
                else{
                    print("Failed to create...")
                }
            }
        }
    }
    if s == 0{
        print("Failed to find Spacecraft \(model!)...")
    }
case 7:
    var s = 0
    var p = 0
    print("Enter model of Spacecraft: ")
    let model = readLine()

    for spacecraft in spacecrafts{
```

```
            if spacecraft.getModel() == model{
                s += 1
                print("Enter model of solar panel: ")
                let solarModel = readLine()
                for panel in solarpanels {
                    if panel.getModel() == solarModel{
                        p += 1
                        spacecraft.addPanel(panel: panel)
                        print("Successfully attached \(solarModel!) to \(model!)")
                    }
                }
                if p == 0{
                    print("Failed to find a Solar Panel with name \(solarModel!)...\
                    let ans = readLine()?.lowercased()
                    if ans == "y"{
                        solarpanels.append(spacecraft.addPanel(solarModel!))
                        print("Success!")
                    }
                    else if ans == "n"{
                        print("As you wish")
                    }
                    else{
                        print("Failed to create...")
                    }
                }
            }
        }
        if s == 0{
            print("Failed to find Spacecraft \(model!)...")
        }

    case 8:
        var p = 0
        print("Enter name of the person: ")
        let name = readLine()
        for person in people {
            if person.getName() == name{
                p += 1
                print("Enter state: ")
                let state = readLine()
                if state == "awake" || state == "on"{
                    person.wakeUp()
                }
                else if state == "asleep" || state == "ofa"{
                    person.sleep()
                }
                else{
                    print("Something went wrong")
                }
            }
```

```
    }
    if p == 0{
        print("Failed to find a person with name \(name!)...)")
    }
case 9:
    var s = 0
    var e = 0
    print("Enter model of Spacecraft: ")
    let model = readLine()

    for spacecraft in spacecrafts{
        if spacecraft.getModel() == model{
            s += 1
            if spacecraft.amountOfEngineers == 0 {
                print("Failed to find an engineer on board of \(model!)")
            }
            else{
                e += 1
                for person in spacecraft.personnel!{
                    if person.proffesion == .engineer{
                        if e == 1{
                            print("Enter state: ")
                            let state = readLine()
                            if state == "on"{
                                let temp = Engineer(person: person)
                                temp.setEngineState(engine: spacecraft.engine, s
                                print("The engine is on")
                            }
                            else if state == "off"{
                                let temp = Engineer(person: person)
                                temp.setEngineState(engine: spacecraft.engine, s
                                print("The engine is off")
                            }
                            else{
                                print("Error : wrong state value")
                            }
                        }
                    }
                }
            }
        }
    }
    if s == 0{
        print("Failed to find Spacecraft \(model!)...")
    }
case 10:
    var s = 0
    var e = 0
    print("Enter model of Spacecraft: ")
    let model = readLine()
```

```
    for spacecraft in spacecrafts{
        if spacecraft.getModel() == model{
            s += 1
            if spacecraft.amountOfScientists == 0 {
                print("Failed to find an scientist on board of \(model!)")
            }
            else{
                e += 1
                for person in spacecraft.personnel!{
                    if person.proffesion == .scientist{
                        if e == 1{
                            print("Enter state: ")
                            let state = readLine()
                            if state == "on"{
                                let temp = Scientist(person: person)
                                if let panels = spacecraft.solarPanels{
                                    for panel in panels{
                                        temp.setPanelState(panel: panel, state:
                                        print("Panel \(panel.getModel()) is on")
                                    }
                                }
                            }
                            else if state == "off"{
                                let temp = Scientist(person: person)
                                if let panels = spacecraft.solarPanels{
                                    for panel in panels{
                                        temp.setPanelState(panel: panel, state:
                                        print("Panel \(panel.getModel()) is off"
                                    }
                                }
                            }
                            else{
                                print("Error : wrong state value")
                            }
                        }
                    }
                }
            }
        }
        if s == 0{
            print("Failed to find Spacecraft \(model!)...")
        }
    case 11:
        print("Enter model of Spacecraft: ")
        let model = readLine()

        for spacecraft in spacecrafts{
            if spacecraft.getModel() == model{
```

```
                    spacecraft.captain?.startFlight(spacecraft: spacecraft)
                }
            }
        case 12:
            print("Warning: the captain must be either engineer or scientist!")
            var s = 0
            var p = 0
            print("Enter model of Spacecraft: ")
            let model = readLine()

            for spacecraft in spacecrafts{
                s += 1
                if spacecraft.getModel() == model{
                    print("Enter name of the person: ")
                    let name = readLine()
                    for person in people {
                        if person.getName() == name{
                            p += 1
                            if spacecraft.getModel() == person.occupation?.getModel(){
                                spacecraft.captain = person
                                print("Successfully chaged captain to \(name!)")
                            }
                            else{
                                print("Procces not authorised")
                            }
                        }
                    }
                    if p == 0{
                        print("Failed to find a person with name \(name!)...")
                        }
                    }
                }

            if s == 0{
                print("Failed to find Spacecraft \(model!)...")
            }

        case 13:

            print("Exiting the program")
            state = false
        default:
            state = false
        }
        print("\nWhat's next?\n")
    }
    else{
        print("Wrong input, try again")
    }
}
```

3.2 Spacecraft.swift

```
//
//  Spacecraft.swift
//  Spacecraft
//
//  Created by Nastya Bekesheva on 08.04.2022.
//

import Foundation

class Spacecraft{

    var engine: Engine
    private var model: String
    var captain: Personnel?
    var personnel: [Personnel]?
    var solarPanels: [SolarPanel]?
    var amountOfScientists: Int = 0
    var amountOfEngineers: Int = 0

    init(model: String, engine: String, captain: Personnel?, personnel: [Personnel]?){
        self.model = model
        self.engine = Engine(engine)
        self.captain = captain
        self.personnel = personnel
        self.solarPanels = nil
        self.personnel = [captain!]
        switch captain!.proffesion{
        case .engineer:
            amountOfEngineers += 1
        case .scientist:
            amountOfScientists += 1
        case .none:
            break
        }
        captain!.occupation = self
    }

    deinit{
        print("Bye")
    }

    func addPersonnel(_ name: String) -> Personnel{
        let temp = Personnel(name)
        if let _ = personnel {
            personnel?.append(temp)
        }
```

```
    else{
        personnel = [temp]
    }
    temp.occupation = self

    return temp
}
func addPersonnel(person: Personnel){
    if let _ = personnel {
        personnel?.append(person)
    }
    else{
        personnel = [person]
    }
    switch person.proffesion{
    case .engineer:
        amountOfEngineers += 1
    case .scientist:
        amountOfScientists += 1
    case .none:
        break
    }
    person.occupation = self

}

func addPanel(_ name: String) -> SolarPanel{
    let temp = SolarPanel(name)
    if let _ = solarPanels {
        solarPanels?.append(temp)
    }
    else{
        solarPanels = [temp]
    }

    return temp
}
func addPanel(panel: SolarPanel){
    if let _ = solarPanels {
        solarPanels?.append(panel)
    }
    else{
        solarPanels = [panel]
    }
}

func getModel() -> String{
    return self.model
}
```

```
}
```

3.3 Engine.swift

```
//
//  Engine.swift
//  Spacecraft
//
//  Created by Nastya Bekesheva on 08.04.2022.
//

import Foundation

class Engine{

    private let model: String
    var state: State

    enum State{
        case on, off
    }

    init(_ model: String){
        self.model = model
        self.state = .off
    }
}
```

3.4 SolarPanels

```
//
//  SolarPanels.swift
//  Spacecraft
//
//  Created by Nastya Bekesheva on 08.04.2022.
//

import Foundation

class SolarPanel{

    private let model: String
    var state: State

    enum State{
        case on, off
    }

    init(_ model: String){
        self.model = model
```

```
        self.state = .off
    }

    func getModel() -> String{
        return self.model
    }

}
```

3.5 Personnel

```
//
//  Personnel.swift
//  Spacecraft
//
//  Created by Nastya Bekesheva on 08.04.2022.
//

import Foundation

class Personnel{

    private let name: String
    var state: State
    var proffesion: Proffesion?
    var occupation: Spacecraft?

    enum State{
        case asleep, awake
    }

    enum Proffesion{
        case scientist, engineer
    }

    init(){
        self.name = ""
        self.state = .asleep
        self.proffesion = nil
        self.occupation = nil
    }

    init(_ name: String){
        self.name = name
        self.state = .awake
        self.proffesion = nil
        self.occupation = nil
    }

    func startFlight(spacecraft: Spacecraft){
```

```
if self.occupation?.getModel() == spacecraft.getModel(){
    print("Preparing for flight...")
    if spacecraft.amountOfEngineers != 0 && spacecraft.amountOfScientists != 0{
        if let captain = spacecraft.captain {
            switch captain.proffesion{
            case .engineer:
                let updatedCaptain = Engineer(person: captain)
                if updatedCaptain.checkEngine(engine: occupation!.engine)!{
                    if let solarPanels = spacecraft.solarPanels {
                        var scientist = Scientist()
                        for person in spacecraft.personnel!{
                            if person.proffesion == .scientist{
                                scientist = Scientist(person: person)
                            }
                        }
                        var panelsState = [Bool]()
                        for panel in occupation!.solarPanels!{
                            panelsState.append(scientist.checkPanels(panel: pane
                        }
                        if !panelsState.contains(false){
                            print("Ready to fly!\nSetting off...\nSuccess, you'r
                        }
                        else{
                            print("You should turn your pannels on")
                            print("Ready to fly!\nSetting off...\nSuccess, you'r
                        }
                    }
                    else{
                        print("Cannot fly without solar panels")
                    }

                }
                else{
                    print("Please turn the engine on")
                }
            case .scientist:
                let updatedCaptain = Scientist(person: captain)
                var engineer = Engineer()
                for person in spacecraft.personnel!{
                    if person.proffesion == .engineer{
                        engineer = Engineer(person: person)
                        if engineer.checkEngine(engine: occupation!.engine)!{
                            if let solarPanels = spacecraft.solarPanels {
                                var panelsState = [Bool]()
                                for panel in occupation!.solarPanels!{
                                    panelsState.append(updatedCaptain.checkPanel
                                }
                                if panelsState.contains(true){
                                    print("Ready to fly!\nSetting off...\nSucces
                                }
```

```
                                    else{
                                        print("You should turn your pannels on")
                                        print("Ready to fly!\nSetting off...\nSucces
                                    }
                                }
                                else{
                                    print("Cannot fly without solar panels")
                                }
                            }
                            else{
                                print("Cannot fly without engine")
                            }
                        }
                    }


                case .none:
                    print("Imposter on board")
                }
            }
        }
        else if spacecraft.amountOfEngineers == 0{
            print("Cannot fly without engineers")
        }
        else if spacecraft.amountOfScientists == 0{
            print("Cannot fly without scientists")
        }
        else{
            print("Cannot fly without personnel")
        }
    }
    else{
        print("Procces not authorised")
    }
}

func wakeUp(){
    self.state = .awake
    print("\(self.name) is now awake")
}

func sleep(){
    self.state = .asleep
    print("\(self.name) is now sleeping")
}

func getName() -> String{
    return self.name
}
```

```
    deinit{
        print("Deleteing unnecessary object")
    }
}
```

3.6 Scientist

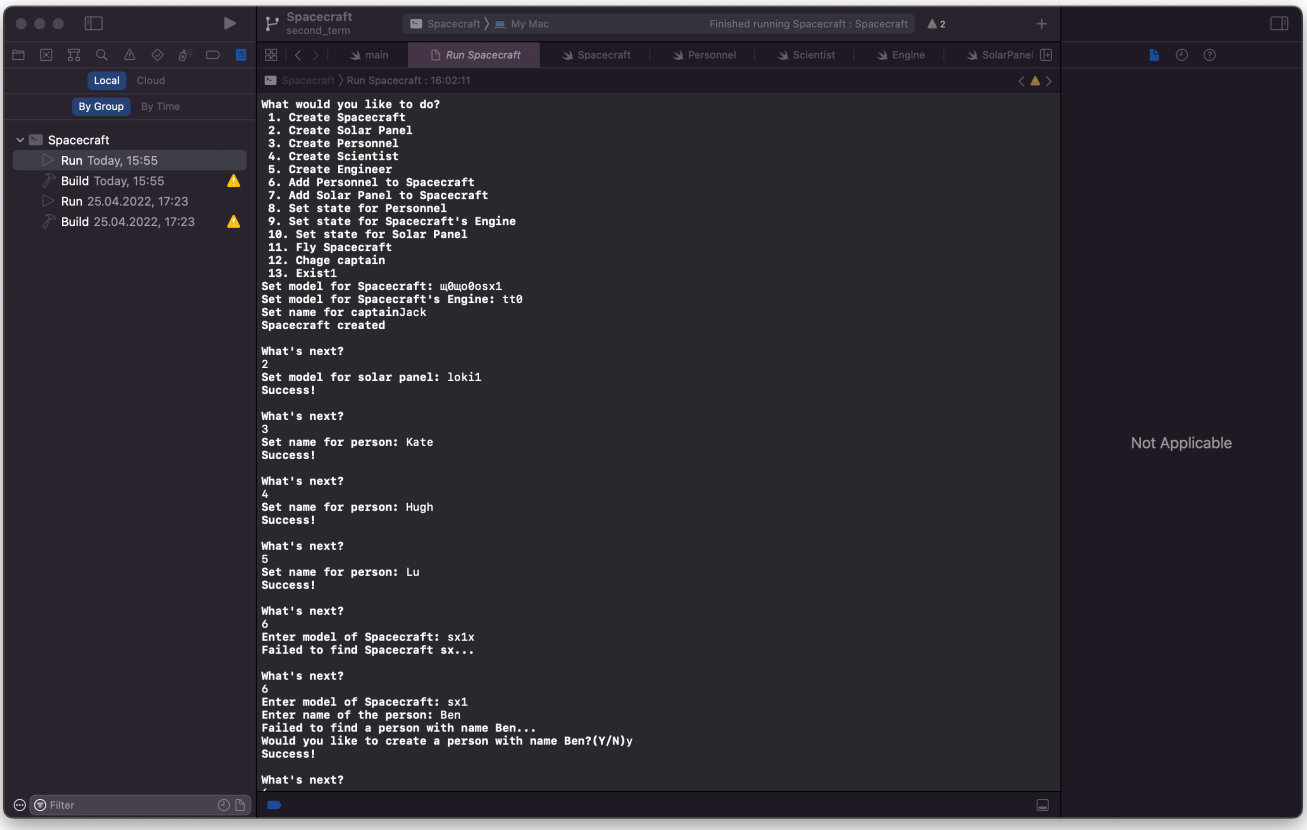
```
//
//  Scientist.swift
//  Spacecraft
//
//  Created by Nastya Bekesheva on 08.04.2022.
//

import Foundation

class Scientist: Personnel{

    override init(){super.init()}

    override init(_ name: String){
        super.init(name)
        proffesion = .scientist
    }

    init(person: Personnel){
        super.init(person.getName())
        proffesion = .scientist
    }


    func checkPanels(panel: SolarPanel) -> Bool?{

        switch self.state{
        case .awake:
            switch panel.state{
            case .on:
                return true
            case .off:
                return false
            }
        case .asleep:
            print("Cannot check panels while sleeping")
            return nil
        }
    }

    func setPanelState(panel: SolarPanel, state: SolarPanel.State){
        panel.state = state
    }
```

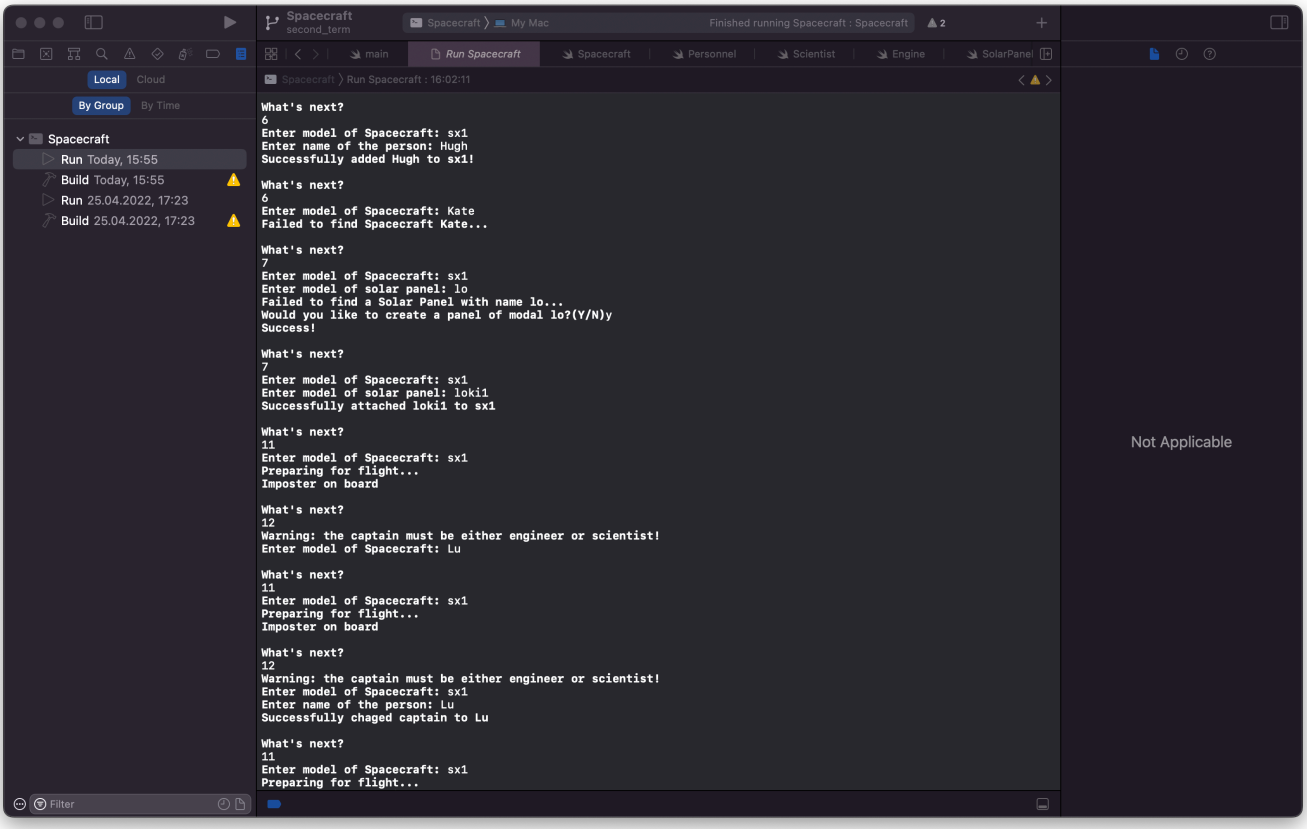
```
}
```

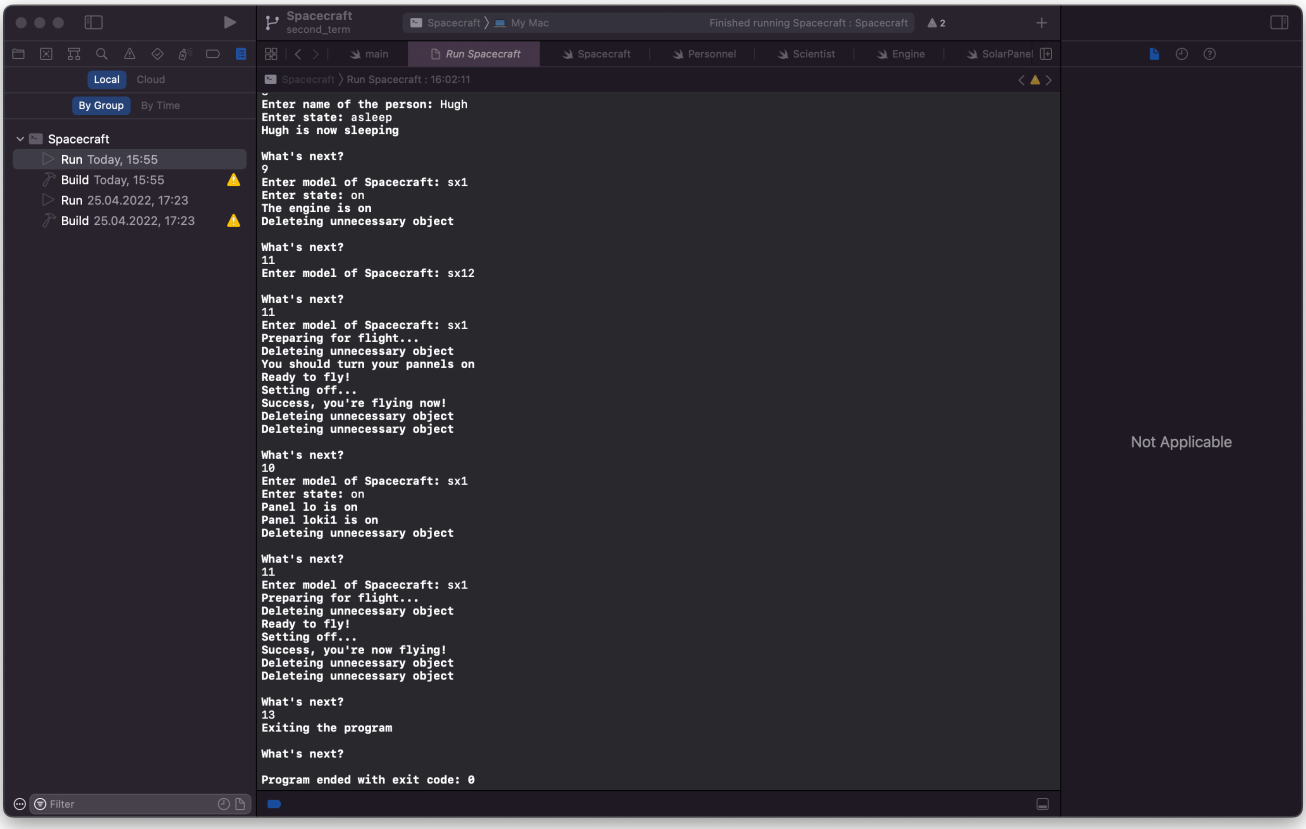
3.7 Engineer

```
//
//  Engineer.swift
//  Spacecraft
//
//  Created by Nastya Bekesheva on 08.04.2022.
//

import Foundation

class Engineer: Personnel{

    override init(){super.init()}

    override init(_ name: String){
        super.init(name)
        proffesion = .engineer
    }

    init(person: Personnel){
        super.init(person.getName())
        proffesion = .scientist
    }

    func checkEngine(engine: Engine) -> Bool?{
        switch self.state{
        case .awake:
            switch engine.state{
            case .on:
                return true
            case .off:
                return false
            }
        case .asleep:
            print("Cannot check engine while sleeping")
            return nil
        }
    }

    func setEngineState(engine: Engine, state: Engine.State){
        engine.state = state
    }

}
```

4 Скріни виконання роботи






Spacecraft
second_term
Spacecraft 〉 My Mac
Finished running Spacecraft : Spacecraft
Local
Cloud
By Group
By Time
Spacecraft
Run Today, 15:55
Build Today, 15:55
Run 25.04.2022, 17:23
Build 25.04.2022, 17:23
main
Run Spacecraft
Spacecraft
Personnel
Scientist
Engine
SolarPanel
Spacecraft 〉 Run Spacecraft : 16:02:11
Enter name of the person: Hugh
Enter state: asleep
Hugh is now sleeping
What's next?
9
Enter model of Spacecraft: sx1
Enter state: on
The engine is on
Deleteing unnecessary object
What's next?
11
Enter model of Spacecraft: sx12
What's next?
11
Enter model of Spacecraft: sx1
Preparing for flight...
Deleteing unnecessary object
You should turn your pannels on
Ready to fly!
Setting off...
Success, you're flying now!
Deleteing unnecessary object
Deleteing unnecessary object
What's next?
10
Enter model of Spacecraft: sx1
Enter state: on
Panel lo is on
Panel loki1 is on
Deleteing unnecessary object
What's next?
11
Enter model of Spacecraft: sx1
Preparing for flight...
Deleteing unnecessary object
Ready to fly!
Setting off...
Success, you're now flying!
Deleteing unnecessary object
Deleteing unnecessary object
What's next?
13
Exiting the program
What's next?
Program ended with exit code: 0
Filter
Not Applicable