

# Homework 4

## Write up

Brian Kennedy  
Pete Koehn  
Theodore Lindsey

November 19, 2014



# Contents

0.1	Documentation . . . . .	4
0.2	hw04.py . . . . .	4
0.3	cController.py . . . . .	4
0.4	cView.py . . . . .	5
0.5	cModel.py . . . . .	7
0.6	Diagrams . . . . .	11

## 0.1 Documentation

Files needed in order to run:

- `__init__.py`
- `cController.py`
- `cModel.py`
- `cView.py`
- `hw04.py`

To run, place all files in the same directory, then execute the command "python hw04.py" from the command prompt or command line.

## 0.2 hw04.py

---

```
from cController import Controller
```

```
Controller()
```

---

## 0.3 cController.py

---

```
class Controller:
    def __init__(self):
        from cView import View
        from cModel import Model

        model = Model()
        view = View()

        while True:
            action = view.menu()

            if action == "quit":
                quit()
            elif action[0] == "r":
                self.RemoveFromOrder(model, int(action[1:]))
            elif action[0] == "a":
                self.AddToOrder(model, int(action[1:]))
            elif action[0] == "d":
                self.DisplayBook(model, view, action[1:])
```

```

        elif action == "order":
            self.GetOrderList(model, view)
        elif action == "cost":
            self.GetOrderCost(model, view)

    def DisplayBook(self, model, view, genre):
        BookList = model.GetBookList(genre)
        view.GeneralDisplay(BookList)

    def AddToOrder(self, model, BookID):
        model.AddToOrder(BookID)
        return "Added_book_#" + str(BookID) + "_to_the_order."

    def RemoveFromOrder(self, model, BookID):
        model.RemoveFromOrder(BookID)
        return "Removed_book_#" + str(BookID) + "_from_the_order."

    def GetOrderCost(self, model, view):
        cost = model.CalculateOrderCost()
        view.GeneralDisplay(cost)

    def GetOrderList(self, model, view):
        OrderList = model.GetOrder()
        view.GeneralDisplay(OrderList)

```

---

## 0.4 cView.py

---

```

class View:
    #--Display Books--
    def DisplayBooks(self):
        print("Select_genre:")
        print("1. ....Science_Fiction")
        print("2. ....Travel")
        print("3. ....Software_Engineering")
        uchoice = input("Please_input_the_number_of_your_choice:_")
        )
        print(uchoice)
        print()

        if uchoice == "1":
            return "dSciFi"
        elif uchoice == "2":

```

```

        return "dTravel"
    elif uchoice == "3":
        return "dSoftware"

#—Add Book to Order—
def AddToOrder(self):
    uchoice = input("Please input BookID: ")
    if (1 <= int(uchoice) <= 60):
        return "a" + str(uchoice)
    else:
        print("Invalid BookID, please try again.")
        return self.AddToOrder()

#^ this will pass bookid to other thing

#—Remove Book from Order—
def RemoveFromOrder(self):
    uchoice = input("Please input BookID: ")
    if (1 <= int(uchoice) <= 60):
        return "r" + str(uchoice)
    else:
        print("Invalid BookID, please try again.")
        return self.RemoveFromOrder()

def GeneralDisplay(self, array):
    for entry in array:
        print(entry)

#—menu—
def menu(self):
    print()
    print("=====")
    print(" Available actions:")
    print(" 1. .... Display Books")
    print(" 2. .... Display Current Order")
    print(" 3. .... Add Book to Order")
    print(" 4. .... Remove Book from Order")
    print(" 5. .... Calculate Order Cost")
    print(" 6. .... Quit")
    uchoice = input("Please input the number of your choice: ")
    )
    print("")
    if uchoice == '1':
        return self.DisplayBooks()
    elif uchoice == '2':

```

```

        return "order"
    elif uchoice == '3':
        return self.AddToOrder()
    elif uchoice == '4':
        return self.RemoveFromOrder()
    elif uchoice == '5':
        return "cost"#self.DisplayOrderCost()
    elif uchoice == '6':
        return "quit"
    else:
        print("Invalid input, please try again.")
        return "invalid"

```

---

## 0.5 cModel.py

---

```

#!/usr/bin/python
class Model:
    def __init__(self):
        self.CurrentOrder = []
        self.cost = 0

        self.titles = [0] * 61
        self.authors = [0] * 61

    # Science Fiction titles, $50
    self.titles[1] = 'Dune_[S1]'
    self.titles[2] = 'Ender\'s_Game_[S1]'
    self.titles[3] = 'The_Foundation_Trilogy'
    self.titles[4] = 'Hitch_Hiker\'s_Guide_to_the_Galaxy_[S1]'
    self.titles[5] = '1984'
    self.titles[6] = 'Stranger_in_a_Strange_Land'
    self.titles[7] = 'Fahrenheit_451'
    self.titles[8] = '2001:_A_Space_Odyssey'
    self.titles[9] = 'Do_Androids_Dream_of_Electric_Sheep?'
    self.titles[10] = 'Neuromancer_[S1]'
    self.titles[11] = '[C]_I,_Robot'
    self.titles[12] = 'Starship_Troopers'
    self.titles[13] = 'Ringworld_[S1]'
    self.titles[14] = 'Rendezvous_With_Rama'
    self.titles[15] = 'Hyperion_[S1]'
    self.titles[16] = 'Brave_New_World'
    self.titles[17] = 'The_Forever_War'
    self.titles[18] = 'The_Time_Machine'

```

```
self.titles[19] = 'Childhood\'s_End'
self.titles[20] = 'The_Moon_is_a_Harsh_Mistress'
```

*# Travel titles , \$40*

```
self.titles[21] = 'A_Dragon_Apparent'
self.titles[22] = 'A_House_in_Bali'
self.titles[23] = 'A_Moveable_Feast'
self.titles[24] = 'A_Short_Walk_in_the_Hindu_Kush'
self.titles[25] = 'A_Time_of_Gifts'
self.titles[26] = 'A_Turn_in_the_South'
self.titles[27] = 'A_Walk_in_the_Woods'
self.titles[28] = 'A_Winter_in_Arabia'
self.titles[29] = 'Among_the_Russians'
self.titles[30] = 'An_Area_of_Darkness'
self.titles[31] = 'Arabian_Sands'
self.titles[32] = 'Arctic_Dreams'
self.titles[33] = 'The_Art_of_Travel'
self.titles[34] = 'As_I_Walked_Out_One_Midsummer_Morning'
self.titles[35] = 'Baghdad_Without_a_Map'
self.titles[36] = 'Balkan_Ghosts'
self.titles[37] = 'Beyond_Euphrates'
self.titles[38] = 'The_Bird_Man_and_the_Lap_Dancer'
self.titles[39] = 'Bitter_Lemons_of_Cyprus'
self.titles[40] = 'Black_Lamb_and_Grey_Falcon'
```

*# Software Engineering titles , \$100*

```
self.titles[41] = 'Code_Complete:_A_Handbook_of_Software_
    Construction'
self.titles[42] = 'Head_First_Design_Patterns'
self.titles[43] = 'Rapid_Development'
self.titles[44] = 'Design_Patterns:_Elements_of_Reusable_
    Object-Oriented_Software'
self.titles[45] = 'Cryptography:_Protocols,_Algorithms,_
    and_Source_Code'
self.titles[46] = 'Agile_Software_Development:_Principles,
    Patterns_and_Practices'
self.titles[47] = 'Joel_on_Software'
self.titles[48] = 'Peopleware:_Productive_Projects_and_
    Teams'
self.titles[49] = 'The_Mythical_Man-Month,_Anniversary_
    Edition'
self.titles[50] = 'Refactoring:_Improving_the_Design_of_
    Existing_Code'
self.titles[51] = 'Agile_Estimating_and_Planning'
self.titles[52] = 'Writing_Effective_Use_Cases'
```



```

self.titles[53] = 'Object-Oriented_Software_Construction'
self.titles[54] = 'Software_Estimation:_Demystifying_the_
    Black_Art'
self.titles[55] = 'User_Stories_Applied:_For_Agile_
    Software_Development'
self.titles[56] = 'The_Art_of_Computer_Programming'
self.titles[57] = 'Patterns_of_Enterprise_Application_
    Architecture'
self.titles[58] = 'Mastering_Regular_Expressions'
self.titles[59] = 'The_Pragmatic_Programmer'
self.titles[60] = 'Software_Requirements'

```

*# Science Fiction titles , \$50*

```

self.authors[1] = 'Frank_Herbert'
self.authors[2] = 'Orson_Scott_Card'
self.authors[3] = 'Isaac_Asimov'
self.authors[4] = 'Douglas_Adams'
self.authors[5] = 'George_Orwell'
self.authors[6] = 'Robert_A_Heinlein'
self.authors[7] = 'Ray_Bradbury'
self.authors[8] = 'Arthur_C_Clarke'
self.authors[9] = 'Philip_K_Dick'
self.authors[10] = 'William_Gibson'
self.authors[11] = 'Isaac_Asimov'
self.authors[12] = 'Robert_A_Heinlein'
self.authors[13] = 'Larry_Niven'
self.authors[14] = 'Arthur_C_Clarke'
self.authors[15] = 'Dan_Simmons'
self.authors[16] = 'Aldous_Huxley'
self.authors[17] = 'Joe_Haldeman'
self.authors[18] = 'H_G_Wells'
self.authors[19] = 'Arthur_C_Clarke'
self.authors[20] = 'Robert_A_Heinlein'

```

*# Travel titles , \$40*

```

self.authors[21] = 'Norman_Lewis'
self.authors[22] = 'Colin_McPhee'
self.authors[23] = 'Ernest_Hemingway'
self.authors[24] = 'Eric_Newby'
self.authors[25] = 'Patrick_Leigh_Fermor'
self.authors[26] = 'V.S._Naipaul'
self.authors[27] = 'Bill_Bryson'
self.authors[28] = 'Freya_Stark'
self.authors[29] = 'Colin_Thubron'
self.authors[30] = 'V.S._Naipaul'

```

```

self.authors[31] = 'Wilfred_Thesiger'
self.authors[32] = 'Barry_Lopez'
self.authors[33] = 'Alain_de_Botton'
self.authors[34] = 'Laurie_Lee'
self.authors[35] = 'Tony_Horwitz'
self.authors[36] = 'Robert_D._Kaplan'
self.authors[37] = 'Freya_Stark'
self.authors[38] = 'Eric_Hansen'
self.authors[39] = 'Lawrence_Durrell'
self.authors[40] = 'Rebecca_West'

# Software Engineering titles, $100
self.authors[41] = 'Steve_McConnell'
self.authors[42] = 'Elisabeth_Freeman'
self.authors[43] = 'Steve_McConnell'
self.authors[44] = 'Erich_Gamma'
self.authors[45] = 'Bruce_Schneier'
self.authors[46] = 'Robert_C._Martin'
self.authors[47] = 'Joel_Spolsky'
self.authors[48] = 'Tom_DeMarco'
self.authors[49] = 'Frederick_P._Brooks'
self.authors[50] = 'Martin_Fowler'
self.authors[51] = 'Mike_Cohn'
self.authors[52] = 'Alistair_Cockburn'
self.authors[53] = 'Bertrand_Meyer'
self.authors[54] = 'Steve_McConnell'
self.authors[55] = 'Mike_Cohn'
self.authors[56] = 'Donald_E._Knuth'
self.authors[57] = 'Martin_Fowler'
self.authors[58] = 'Jeffrey_Friedl'
self.authors[59] = 'Andrew_Hunt'
self.authors[60] = 'Karl_E._Wiegers'

def GetBookList(self, genre):
    BookList = ["ID\tAuthor\t\t\t\tTitle"]
    if genre == "SciFi":
        for BookID in range(1,20+1):
            BookList.append(str(BookID) + "\t" + self.authors[
                BookID] + "\t\t\t" + self.titles[BookID])
    elif genre == "Travel":
        for BookID in range(21,40+1):
            BookList.append(str(BookID) + "\t" + self.authors[
                BookID] + "\t\t\t" + self.titles[BookID])
    elif genre == "Software":
        for BookID in range(41,60+1):

```

```

        BookList.append(str(BookID) + "\t" + self.authors[
            BookID] + "\t\t\t" + self.titles[BookID])
    return BookList

def GetOrder(self):
    OrderList = ["Current_order_contents:"]
    OrderList.append("ID\tAuthor\t\t\tTitle")
    for BookID in self.CurrentOrder:
        OrderList.append(str(BookID) + "\t" + self.authors[
            BookID] + "\t\t\t" + self.titles[BookID])
    return OrderList

def AddToOrder(self, BookID):
    self.CurrentOrder.append(BookID)
    self.CurrentOrder.sort()

def RemoveFromOrder(self, BookID):
    try:
        self.CurrentOrder.remove(BookID)
    except:
        print("Book_ID_" + str(BookID) + "_not_in_current_"
            order")

def CalculateOrderCost(self):
    self.cost = 0
    for i in self.CurrentOrder:

        #print("Cost: " + str(self.cost))
        #print("BookID: " + str(i))
        #print()
        if i < 21:
            self.cost += 50
        elif i < 41:
            self.cost += 40
        else:
            self.cost += 100
    return ["Order_cost_is_$" + str(self.cost)]

```

---

## 0.6 Diagrams

Figure 1: Sequence Diagram

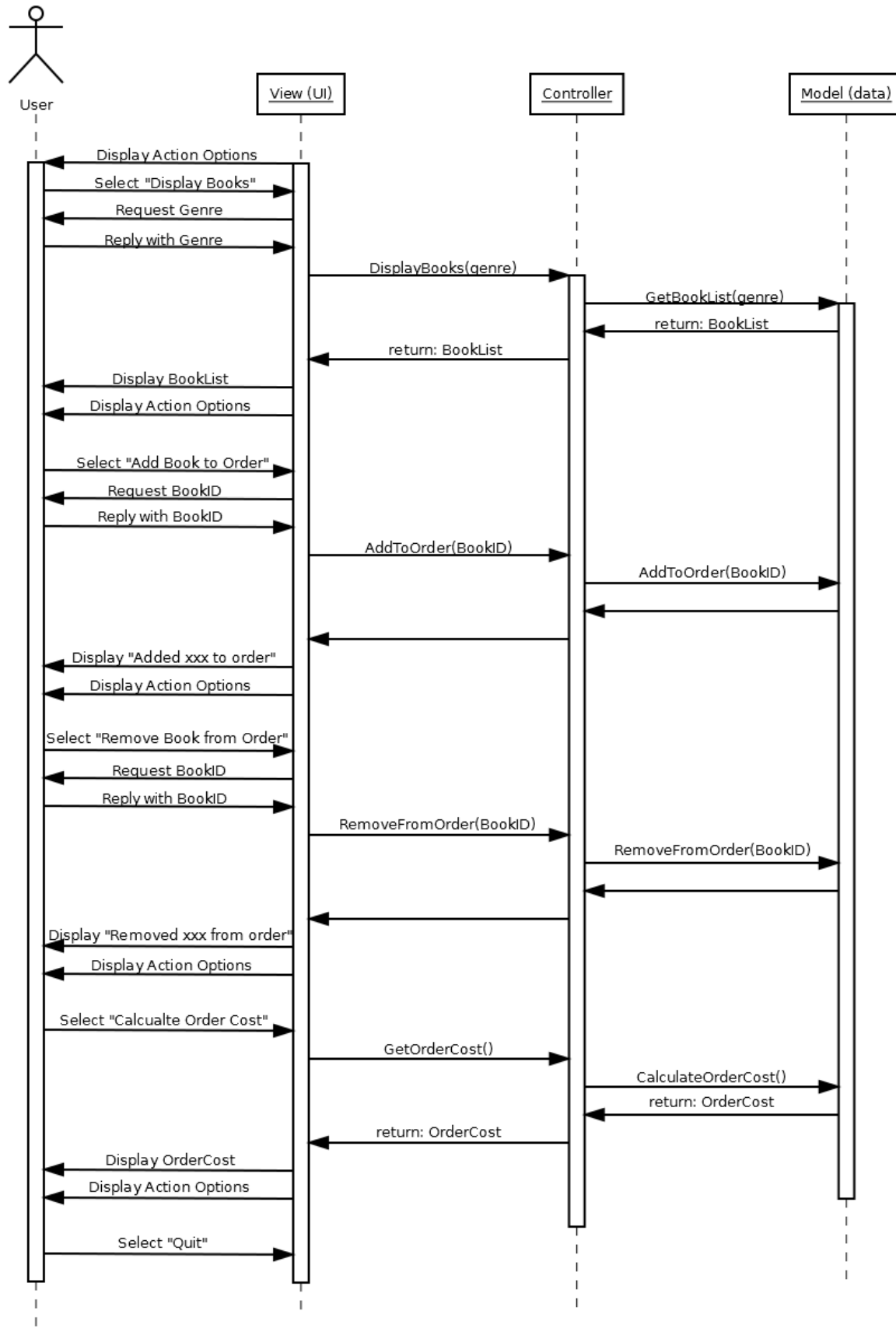


Figure 2: Class Diagram

