Deakin University

Object Oriented Development

OnTrack Submission

Bucket Sort

Submitted By:
Connor Gent
gentco
2021/05/29 15:10

Tutor:
Nayyar Zaidi

Outcome	Weight
Evaluate Code	♦♦♦♦◊
Principles	♦♦♦♦◊
Build Programs	♦♦♦♦◊
Design	♦♦♦♦◊
Justify	♦♦♦♦◊

Bucket sort task 3.4d which introduces us to an algorithm that sorts data by partitioning the elements of an array into a number of buckets.

May 29, 2021

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Connor_Gent_3._4D_task
{
    class Program
    {
        enum MenuOption
        {
            Withdraw,
            Deposit,
            Print,
            Quit

        }

        static void DoDeposit(Account account)
        {
            Boolean result;
            Console.WriteLine("Enter a deposit amount: ");
            decimal amount = Convert.ToDecimal(Console.ReadLine());
            result = account.Deposit(amount);

            if(result == true)
            {
                Console.WriteLine("Transaction was successful");
            }

            else
            {
                Console.WriteLine("Your transaction was unsuccessful");
            }
        }

        static void DoWithdrawl(Account account)
        {
            Boolean result;
            Console.WriteLine("Enter the amount you want to withdraw: ");
            decimal amount = Convert.ToDecimal(Console.ReadLine());
            result = account.Withdraw(amount);

            if (result == true)
            {
                Console.WriteLine("Your withdrawl was successful");

            }

            else if (result == false)
            {
                Console.WriteLine("Your withdrawl was unsuccesssful");
            }
        }
```

```
54
55          static MenuOption ReadOption()
56          {
57              int? option = null;
58              do
59              {
60                  Console.WriteLine("Please select out of the following");
61                  Console.WriteLine("MENU \n1. WITHDRAW \n2. DEPOSIT \n3. PRINT \n4.
                    ↪  QUIT");
62
63                  try
64                  {
65                      option = Convert.ToInt32(Console.ReadLine());
66                      if (option > 4 || option < 1)
67                      {
68                          option = null;
69                          Console.WriteLine("Please enter a number from 1 to 4");
70                      }
71                  }
72                  catch (FormatException)
73                  {
74                      Console.WriteLine("Invalid input, Must enter a number");
75
76                  }
77              } while (option == null);
78
79              return (MenuOption)option;
80
81          }
82
83          static void DoPrint(Account account)
84          {
85              account.Print();
86          }
87
88
89          static void PrintAccountArray(Account[] accounts)
90          {
91              foreach (Account account in accounts)
92                  account.Print();
93          }
94
95          static void Main(string[] args)
96          {
97              Account[] accounts_Array = new Account[3];
98              accounts_Array[0] = new Account(Convert.ToDecimal(7643.30), "James
                ↪  Harden");
99              accounts_Array[1] = new Account(Convert.ToDecimal(7775.20), "Lebron
                ↪  Jame");
100             accounts_Array[2] = new Account(Convert.ToDecimal(2222.33), "Dustin
                ↪  James");
101             Console.WriteLine("---Accounts before sorting---");
102             PrintAccountArray(accounts_Array);
```

```
103             Console.WriteLine("---Accounts after sorting---");
104             AccountsSorter.Sort(accounts_Array, 3);
105             PrintAccountArray(accounts_Array);
106 
107             Console.WriteLine("----------------------");
108 
109             List<Account> accounts_List = new List<Account>();
110             accounts_List.Add(new Account(Convert.ToDecimal(53422.21), "Sam
                ↪  fanning"));
111             accounts_List.Add(new Account(Convert.ToDecimal(42392.54), "Jesse
                ↪  Hogan"));
112             accounts_List.Add(new Account(Convert.ToDecimal(7654.43), "Issac Que"));
113 
114             Console.WriteLine("***Accounts before sorting***");
115             PrintAccountArray(accounts_List.ToArray());
116 
117             Console.WriteLine("***Accounts after sorting***");
118             AccountsSorter.Sort(accounts_List, 3);
119             PrintAccountArray(accounts_List.ToArray());
120 
121 
122 
123             Account Persone = new Account(200, "Jackson M");
124             Account Perstwo = new Account(300, "Martin D");
125             MenuOption option;
126 
127             do
128             {
129                 option = ReadOption() - 1;
130 
131                 switch (option)
132                 {
133                     case MenuOption.Withdraw:
134                         Console.WriteLine("Withdraw has been selected");
135                         Console.WriteLine("---Jackson's Account---");
136                         DoWithdrawl(Persone);
137                         Console.WriteLine("---Martin's Account---");
138                         DoWithdrawl(Perstwo);
139                         break;
140                     case MenuOption.Deposit:
141                         Console.WriteLine("Deposit has been selcted");
142                         Console.WriteLine("---Jackson's Account---");
143                         DoDeposit(Persone);
144                         Console.WriteLine("---Martin's Account---");
145                         DoDeposit(Perstwo);
146                         break;
147                     case MenuOption.Print:
148                         Console.WriteLine("Print has been selected");
149                         Console.WriteLine("---Jackson's Account---");
150                         DoPrint(Persone);
151                         Console.WriteLine("---Martin---");
152                         DoPrint(Perstwo);
153                         break;
```

```
154                     case MenuOption.Quit:
155                         Console.WriteLine("See youuuuuuu");
156                         break;
157                 }
158             } while (option != MenuOption.Quit);
159         }
160 
161     }
162 }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Connor_Gent_3._4D_task
{
    static class AccountsSorter
    {
        private static decimal MaxBalance (Account[] accounts)
        {
            return accounts.Max(a => a.Balance);
        }



        private static List<Account>[] CreateBuckets(int b)
        {
            List<Account>[] buckets = new List<Account>[b];
            for(int i =0; i<buckets.Length; i++)
            {
                buckets[i] = new List<Account>();
            }
            return buckets;
        }

        private static void DistibuteAccount(Account[] accounts, List<Account>[]
        ↪ buckets)
        {
            decimal maximum = MaxBalance(accounts);
            foreach(Account account in accounts)
            {
                int bucket = (int)(Math.Floor(buckets.Length * account.Balance /
                ↪ maximum));
                if (bucket == buckets.Length)
                    bucket -= 1;
                buckets[bucket].Add(account);
            }
        }

        private static void SortBuckets(List<Account>[] buckets)
        {
            for (int i = 0; i < buckets.Length; i++)
            {
                buckets[i] = buckets[i].OrderBy(a => a.Balance).ToList();
            }
        }

        public static void Sort(Account[] accounts, int b)
        {
            if (accounts == null)
            {
```

```
52                  throw new NullReferenceException("Account must not be null");
53              }
54              if (b <= 1)
55              {
56                      throw new ArgumentOutOfRangeException("Buckets cant be less
                        ↪  than 2");
57              }
58
59              List<Account>[] buckets = CreateBuckets(b);
60              DistibuteAccount(accounts, buckets);
61              SortBuckets(buckets);
62
63              int idx = 0;
64              for(int i = 0; i < buckets.Length; i++)
65              {
66                  foreach(Account account in buckets[i])
67                  {
68                      accounts[idx] = account;
69                      idx++;
70                  }
71
72                  Console.WriteLine();
73              }
74
75          }
76
77          public static void Sort(List<Account> accounts, int b)
78          {
79              if (accounts == null)
80              {
81                  throw new NullReferenceException("Accounts cant be null");
82
83              }
84              Account[] accountsArray = accounts.ToArray();
85              Sort(accountsArray, b);
86              for(int i = 0; i < accounts.Count; i++)
87              {
88                  accounts[i] = accountsArray[i];
89              }
90
91              Console.WriteLine();
92          }
93      }
94  }
95
96
97
98
99
100
```

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Connor_Gent_3._4D_task
{
    public class Account
    {
        private decimal _balance;
        private string _name;

        public string Name { get => _name; }
        public decimal Balance { get => _balance; }

        public Account(decimal balance, string name)
        {
            _name = name;
            if (balance <= 0)
                return;
            _balance = balance;
        }

        public void Print()
        {
            Console.WriteLine("The Holders name is: " + getName() + "\nCurrent
            ↪  Account Balance is " + getBalance());
        }

        public string getName()
        {
            return this._name;
        }

        public decimal getBalance()
        {
            return this._balance;
        }

        public Boolean Deposit(decimal amount)
        {
            if (amount <= 0)
                return false;
            this._balance += amount;
            return true;
        }

        public Boolean Withdraw(decimal amount)
        {
            if (amount > this._balance || amount < 0)
                return false;
            this._balance -= amount;
```

```
53                 return true;
54             }
55
56
57         }
58
59 }
```