CRC Cards Paper Summary

Patwardhan, Indrajeet
Department of Computer Science
California State University, Fullerton
Fullerton, United States
Indrajeet2002@csu.fullerton.edu

***Abstract*—A Laboratory for Teaching Object-Oriented Thinking is an important paper by Kent Beck of Apple Computer, Inc and Ward Cunningham of Wyatt Software Services, Inc. It discusses a new way to design object oriented programs, and its benefits for beginner as well as advanced programmers.**

Keywords—class, responsibility, collaboration, object, system

I. Introduction

CRC Cards are introduced so that beginner programmers can directly experience objects, and advanced programmers can better understand complex designs of object oriented systems.

II. Problem

It is difficult to make beginner programmers shift away from procedural programs and shift toward programs that implement objects. To accomplish this, the object-ness of the material is emphasized to beginner programmers through cards Syntax is not given as much importance because it is relatively easier to learn once the beginners grasp the concepts.

III. Perspective

The design of objects is described by class, responsibilities and collaboration, the same way procedural designs are described by data stores, data flows and processes. The class name describes the object's name in relation to the system. Responsibilities describe what the object can do with respect to the problem to be solved through phrases with a verb. Collaboration describes the relationship of the object to other objects in the system, such as inheritance or composition. It is not always symmetric because object 1 can use object 2 more than object 2 uses object 1.

IV. CRC Cards

CRC cards were developed because they are portable, and they make it easier to visualize the objects in an object oriented system, and how they are related to each other. CRC stands for Class, Responsibility, and Collaboration. The Class section is underlined on the top left of the card, the Responsibility section is below the Class section on the left and the Collaboration section is on the right half of the card. Inheritance and composition can also be shown because if an object extends or has another object then that object is placed closer to the second object due to close collaboration. The goal is to keep an object's description concise, and delegate some of its responsibilities to new objects otherwise. It is important to create only as many objects as necessary at the time so that the system does not become complicated. When using the cards to design a system, the knowns are created first and the unknowns are created based on that. This means the creation of objects does not always follow a pattern in which parent classes are created before child classes. Additionally, physically moving the cards helps to show the execution of the code and how the objects are called upon at various points in the code. CRC cards are used in a three-hour class called "Thinking with Objects" which is for professionals who have programming experience but do not have to program daily for work. The class first explains the concepts and then pairs up the students so that they have an hour to design an automatic banking machine. The class ends by talking about different object oriented programming languages and how they were developed.

V. Technology

The only technology needed to run Cunningham and Beck's method is a stack of 4"x6" index cards, where each index card represents an object and the placement of the cards indicates how the objects depend on each other. This ensures that programmers proficient in different languages can use this method and easily explain code to other programmers because there is no syntax specific to one language. The cards are used in presentations because of their versatility. Also, it allows beginner programmers to understand objects in general and improve their code.

VI. OOPSLA

OOPSLA stands for Object Oriented Programming, Systems, Languages and Applications. It is a conference in North America and was created by pioneers Tom Love,

Adele Goldberg, David Smith and Allen Wirfs-Brock. In November 1986 around 600 people attended the first conference at the Marriott Hotel in Portland, Oregon. Many papers were presented regarding different programming topics by Andrew Black, Ralph Johnson, Ivar Jacobson as well as Ward Cunningham, Kent Beck and others. 3 years later, Cunningham and Beck presented their paper regarding CRC Cards at the 1989 OOPSLA conference at New Orleans, Louisiana. Other important concepts such as the Unified Modeling Language, design patterns, the agile methodologies, dynamic compilation and many others were developed at the conference. Eventually by the 1990s, OOPSLA became a conference for discussing new problems that arose in computing, and how new technologies and techniques could be used to solve them. Although the number of people attending OOPSLA has fluctuated since its creation, it continues to be a place for innovation in computing.

VII. Code Description

An object oriented system for Atm Machine would consist of several objects collaborating with each other. First, an object called Account would have an integer variable called pinNumber and an integer variable called accountMoney. These would store the pin number associated with the account, and the balance of the account, respectively. Account would have a constructor as well as getters and setters for pinNumber and the methods accountMoney and updateBalance. Next, an object called Transaction would have an integer variable called accPin and an integer variable called transAmt. These would allow the transaction to find the account, and user to specify the amount of money to be withdrawn or deposited, respectively. Transaction would have a constructor, getters for accPin and transAmt, and the methods deposit and withdraw. These last two methods would allow the user to deposit or withdraw money from their account. After that, an object called RemoteDataBase would have an ArrayList of Accounts, and a variable of type Transaction. The ArrayList would store the accounts of several users as they use the ATM machine. RemoteDataBase would have an empty constructor and the methods createTransaction, addAccounts and getAccount. These would allow transactions to be made to an account based on the pin number. There would also be the objects Screen and Action which would serve as the interface for the user of the ATM machine. The Screen object would have an integer variable called pinNumber and the method display. The screen object would prompt the user to input their pin number and to press a number for performing an action. For example, if the user inputs 1 for withdrawal, then the input would be sent to the Action object, which would call on the corresponding method of an Account in RemoteDatabase. The Action object would inherit Screen, have a variable of type Dispenser and the method performAction. Finally, an object called Dispenser would just have the method display to indicate that the action was successfully completed.

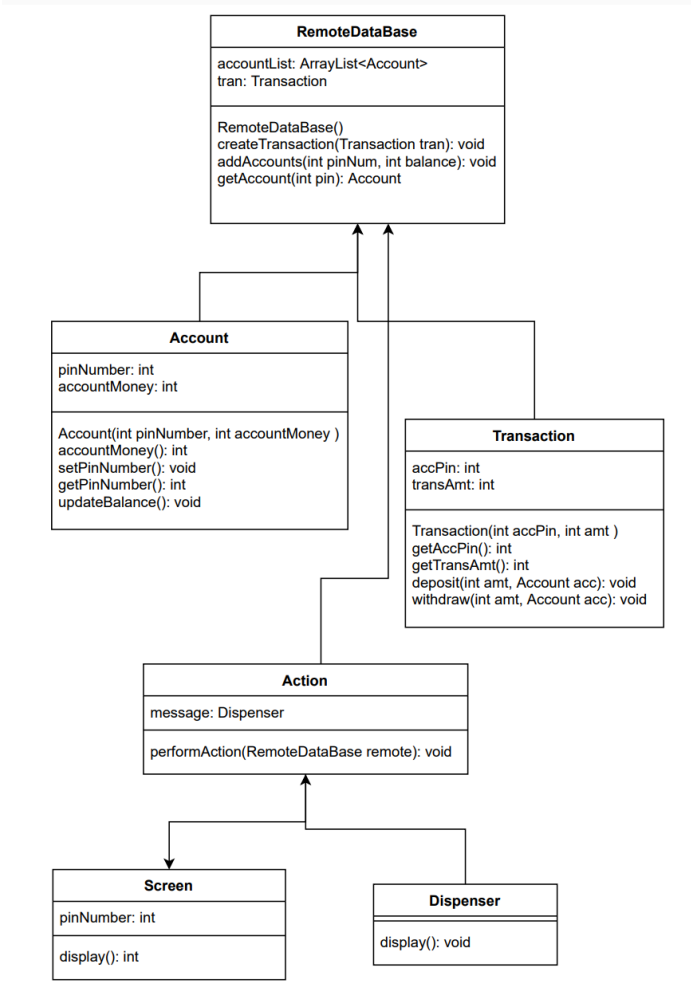
VIII. ATM Machine Implementation

```
package com.company;
import java.util.ArrayList;
import java.util.Scanner;

class AtmMachine {
   public static void main(String[]
args){
       RemoteDataBase data = new
RemoteDataBase();

       data.addAccounts(1234,
12345678);
       data.addAccounts(5555,
99745337);
       data.addAccounts(6789,
21387968);

       Action act = new Action();
       act.display();
       act.performAction(data);
   }
}

class Account {
   int pinNumber;
   int accountMoney;

   public Account(int pinNumber, int
accountMoney){
       this.pinNumber = pinNumber;
       this.accountMoney =
accountMoney;
   }

   public int accountMoney(){
       return accountMoney;
   }

   public void setPinNumber(int
pinNumber){
       this.pinNumber = pinNumber;
   }

   public int getPinNumber(){
       return pinNumber;
   }

   public void updateBalance(int amt){
       accountMoney += amt;
   }
```

```
}

class Transaction {

   int accPin;
   int transAmt;

   public Transaction(int accPin, int
amt){
       this.accPin = accPin;
       this.transAmt = amt;
   }
   public int getAccPin(){
       return accPin;
   }
   public int getTransAmt(){
       return transAmt;
   }
   public void deposit(int amt,
Account acc){
       acc.updateBalance(amt);
   }
   public void withdraw(int amt,
Account acc){
       acc.updateBalance(amt);
   }

}

class RemoteDataBase {

   ArrayList<Account> accountList =
new ArrayList<>();
   Transaction tran;

   public RemoteDataBase(){}
   public void
createTransaction( Transaction tran ){
       this.tran = tran;
       for(Account a : accountList){
           if( a.getPinNumber() ==
tran.getAccPin()){
               if(tran.getTransAmt() >
0){
                   tran.deposit(tran.g
etTransAmt(),a);
                   System.out.println(
"Your new balance is: $" +
a.accountMoney());
               } else{
                   tran.withdraw(tran.
getTransAmt(),a);
                   System.out.println(
"Your new balance is: $" +
a.accountMoney());
               }
           }
       }
   }
   public void addAccounts(int pinNum,
int balance){
       accountList.add( new
Account(pinNum, balance));
   }
   public Account getAccount(int pin){
       for(Account acc : accountList){
           if(acc.getPinNumber() ==
pin){
               return acc;
           }
       }
       return null;
   }
}

class Screen {

   int pinNumber;

   public int display(){
       Scanner sc = new
Scanner(System.in);
       System.out.println("Enter pin
number: ");
       int userIn = sc.nextInt();
       pinNumber = userIn;
       Scanner sc2 = new
Scanner(System.in);
       System.out.println("Welcome to
ATM");
       System.out.println("Press 1 to
make a withdrawal");
       System.out.println("Press 2 to
make a deposit");
       System.out.println("Press 3 to
check balance");
       System.out.println("Press 4 to
change PIN");
       int userIn2 = sc2.nextInt();
       return userIn2;
   }
}

class Action extends Screen {

   Dispenser message = new
Dispenser();
```

```
   public void
performAction(RemoteDataBase remote) {
       Scanner sc2 = new
Scanner(System.in);
       if (super.display() == 1) {
           System.out.println("Enter
the amount you want to withdraw: ");
           int userIn2 =
sc2.nextInt();
           Transaction tran = new
Transaction(pinNumber, userIn2 * -1);
           System.out.println("Withdra
wing: $" + userIn2);
           remote.createTransaction(tr
an);
           message.display();
       }
       else if (super.display() == 2)
{
           System.out.println("Enter
the amount you want to deposit: ");
           int userIn2 =
sc2.nextInt();
           Transaction tran = new
Transaction(pinNumber, userIn2);
           System.out.println("Deposit
ing: $ " + userIn2);
           remote.createTransaction(tr
an);
           message.display();
       }
       else if (super.display() == 3)
{
           System.out.println("Checkin
g balance... ");
           int balance =
remote.getAccount(pinNumber).accountMo
ney();
           System.out.println("Current
ly you have $" + balance + " in your
account.");
           message.display();
       }
       else if (super.display() == 4)
{
           System.out.println("Enter
new PIN: ");
           int userIn2 =
sc2.nextInt();
           System.out.println("Confirm
PIN: ");
           remote.getAccount(pinNumber
).setPinNumber(userIn2);
           message.display();
       }
   }
}

class Dispenser {
   public void display(){
       System.out.println("Action was
successfully completed");
       System.out.println("Thank you,
have a nice day");
   }
}
```

IX. CODE DIAGRAM



Caption

X. Conclusion

In conclusion, CRC cards are very beneficial for beginner programmers because the cards allow them to understand object oriented programming better. Since the concepts of design can be understood faster, they can be better applied to particular languages. The cards are also useful for advanced programmers because they can be used to present an overall system and complete complex designs. The absence of code simplifies the design and makes it less abstract.

References

1. A laboratory for teaching object-oriented thinking. [Online]. Available: http://c2.com/doc/oopsla89/paper.html. [Accessed: 23-Oct-2021].
2. "Oopsla 2008," Redirecting to SPLASH 2010 site... [Online]. Available: http://www.oopsla.org/oopsla-history/. [Accessed: 23-Oct-2021].