CSE2016 Programming Methodology

Week 6: Control Structure: Conditional Statements

Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)

HANYANG UNIVERSITY

Contents

HANYANG UNIVERSITY

Today's Schedule

01. Control Flow and Control Structure

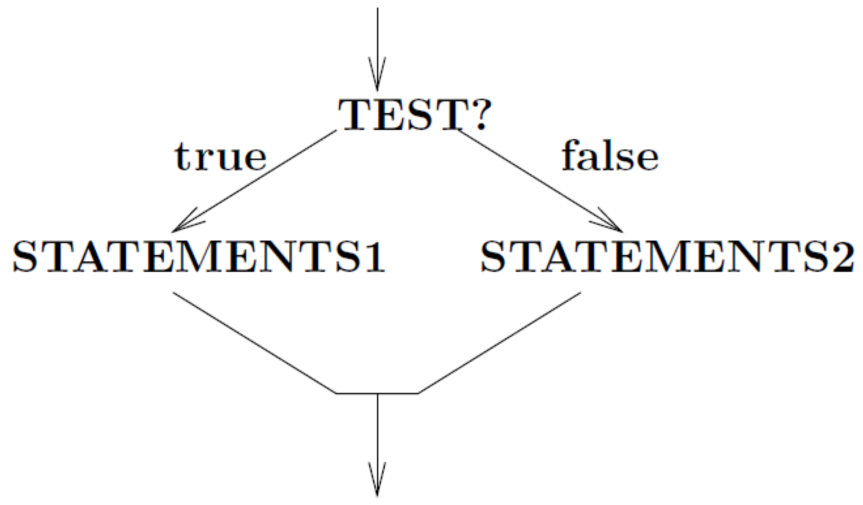
Control

- Control flow
  - The order in which a program's statements execute
- Control structure
  - Sequencing: STATEMENT1; STATEMENT2; ...; STATEMENTn;
  - Method invocation: RECEIVER.METHOD(ARGUMENTS);
  - Conditional statements: If.. Then..
  - …

02. Conditional Control Structure

Grammar

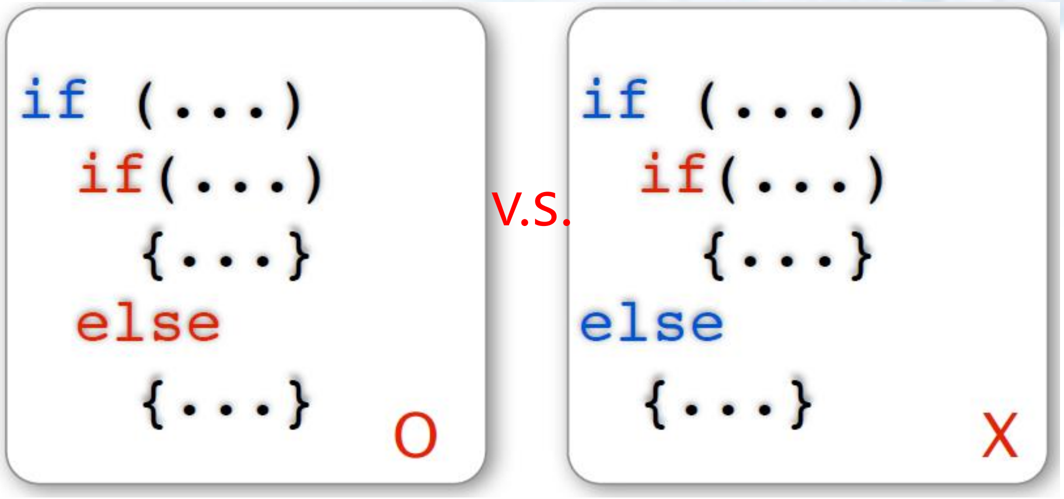
- If (question) statement1 else statement2
- If (question) {statements1} else {statements2}



02. Conditional Control Structure

Note: using "else"

- if (...) if (...) {...} else {...}



```
if (...)
  if(...)
    {...}
  else
    {...}
```

```
if (...)
  if(...)
    {...}
else
  {...}
```

- Let's use braces, "{ ...}"

02. Conditional Control Structure

Example: printPolarity

```
public class printPolarity
{
    public static void main (String[] args)
    {
        int i = -1;
        if (i < 0)
                System.out.print("negative value");
        else
                System.out.println("positive value");
    }
}
```

02. Conditional Control Structure

Example: ConvertHour

```
import javax.swing.*;
public class ConvertHour
{
    public static void main(String[] args)
    {
        int hours = new Integer(JOptionPane.showInputDialog("Input an
        hour.")).intValue();
        if (hours >= 0)
        {
            int seconds = hours * 60 * 60;
            JOptionPane.showMessageDialog(null, hours + " hours are " +seconds
            + " seconds");
        }
        else
        {
            JOptionPane.showMessageDialog(null, "ConvertHours error: negative
            input " + hours);
        }
    }
}
```

02. Conditional Control Structure

HANYANG UNIVERSITY

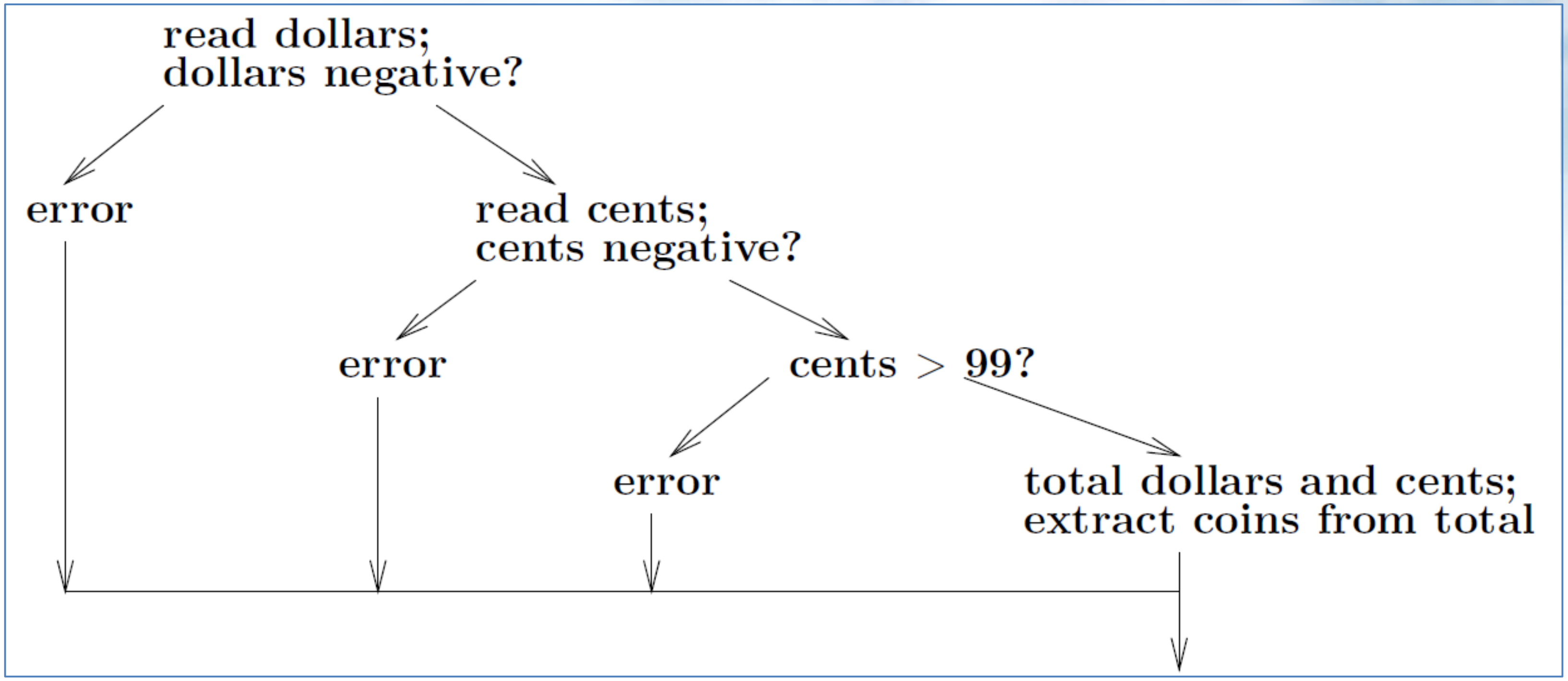
Example: MakeChange

- MakeChange problem
  - Input: dollars (>=0), cents (<100)
  - Output: necessary coins (quarter, dime, nickel, penny)

- Algorithm
  - 1. Read the dollars, an integer
  - 2. If dollars are negative, then generate an error message. Else dollars are nonnegative:
    - (a) If cents are negative, then generate an error message. Else cents are nonnegative:
      - i. If cents are greater than 99, then generate an error message. Else cents are in the range 0..99:
        - A. Compute the total of the dollars and cents
        - B. Extract the appropriate amount of quarter, dime, nickel, and penny coins.

02. Conditional Control Structure

MakeChange



02. Conditional Control Structure

MakeChange

```
public class MakeChange
{
    private static int getInt(String msg)
    {
        return new
    Integer(JOptionPane.showInputDialog(msg)).intValue();
    }

    private static void showError(String msg)
    {
        JOptionPane.showMessageDialog(null, "error: "+msg);
    }

    public static void main(String[] args)
    {
        …
    }
```

Continued..

02. Conditional Control Structure

MakeChange

```
public static void main(String[] args)
{
      int dollars = getInt("dollars?");
      if (dollars < 0)
            showError(dollars + " is a negative value.");
      else
      {
            int cents = getInt("cents?");
            if (cents < 0)
                        showError(cents + “ is a negative value.");
            else
            {
                        if (cents > 99)
                                    ShowError(cents + "is over than 99.");
                        else
                        {
                              int money = (dollars * 100) + cents;
                              String output = "quarters = " + (money / 25);
                              money = money % 25;
                              output = output + "\ndimes = " + (money / 10);
                              money = money % 10;
                              output = output + "\nnickels = " + (money / 5);
                              money = money % 5;
                              output = output + "\npennies = " + money;
                              JOptionPane.showMessageDialog(null, output);
                        }
            }
      }
}
```

03. Relational Operations

HANYANG UNIVERSITY

Boolean Operators

- Conjunction (and)
  - E1 && E2
- Disjunction (or)
  - E1 || E2
- Negation (not)
  - !E

03. Relational Operations

MakeChange

```
public static void main(String[] args)
{
    int dollars = getInt("dollars?");
    int cents = getInt("cents?");
    if (dollars < 0 || cents < 0 || cents > 99)
    {
        showError("invalid input.");
    }
    else
    {
        …
    }
}
```

03. Relational Operations

Notes

- Short-circuit evaluation (or minimal evaluation)
  - Semantics of some Boolean operators in some programming languages in which the second argument is executed or evaluated only if the first argument does not suffice to determine the value of the expression

- Example
  - x<0 || y<0

    ==> -1<0 || y<0

    ==> true || y<0

    ==> true!

04. Uses of Conditionals

twelveHourClock

- Problem
  - Input: hours, minutes in 24-hours clock
    - If input is not proper, return an error message
  - Output: string in 12-hours clock
  - Other considerations
    - 0 hour -> 12 hour
    - 1 mins -> 01 mins

04. Uses of Conditionals

twelveHourClock

```
String answer = "";
if ( hour < 0 || hour > 23 || minute < 0 || minute > 59 )
        answer = "invalid input, " + hour + ":" + minute;
else
{
    if ( hour < 12 )
        answer = answer + "AM ";
    else
        answer = answer + "PM ";

    if ( hour >= 13 )
        answer = answer + (hour - 12);
    else if ( hour == 0 )
        answer = answer + "12";
    else
        answer = answer + hour;

    answer = answer + ":";
    if ( minute < 10 )
        answer = answer + "0";
    answer = answer + minute;
}
```

05. The Switch Statement

Switch

- In many cases,
  - if(x==0) ... else if (x==1) ... else if(x==2) ...
  - Do we have an efficient way?

- “Swtich”
  - A terse style for the nested if-else statements
  - switch(x) { case 0: ... case 1: ... case 2: ... }

05. The Switch Statement

Syntax

```
switch ( EXPRESSION )
  { case VALUE1 : { STATEMENTS1
                    break;
                  }
    case VALUE2 : { STATEMENTS2
                    break;
                  }
             ...

    case VALUEn : { STATEMENTSn
                    break;
                  }
    default : { STATEMENTSn+1 }
  }
```

- EXPRESSION
  - Integer or character variable
- VALUE
  - Integer or character
  - E.g., 2 or ‘a’

05. The Switch Statement

Limitation

- Only values can be used in switch
  - if(x==1) ... else if (x==2) ... else if (x==3) -> switch

- “case y” is not valid
  - if(x==y) cannot be converted to switch statements

- Comparison operators are not used
  - if(x<10) cannot be converted to switch statements

05. The Switch Statement

Note: “break”

- If “break;” is missed in each case, the algorithm may not operate properly

- “break;” may be omitted in some special cases
  - switch(x) {

    case 0:

    case 2:

    … }

- In many cases, it is not recommended

06. Altering Control Flow

Changing the Flow

- Exception
  - If any exception occurs, the program ends
  - Exception messages are printed
- System.exit(0);
  - End the program immediately
- return; or return <statement>;
  - End the program immediately

06. Altering Control Flow

Exception Handing

- Using "throw new RuntimeException(error);"
  - It generates an exception message

```
if (dollars < 0 || cents < 0 || cents > 99)
{
    throw new RuntimeException("invalid input.");
    //showError("invalid input.");
}
```

07. Case Study: Bank Account Manager

HANYANG UNIVERSITY

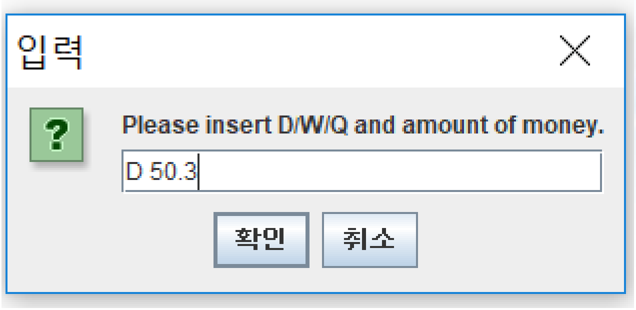
Problem

- Input
  - D, deposit, dollars and cents
  - W, withdraw, dollars and cents
  - Q, quit

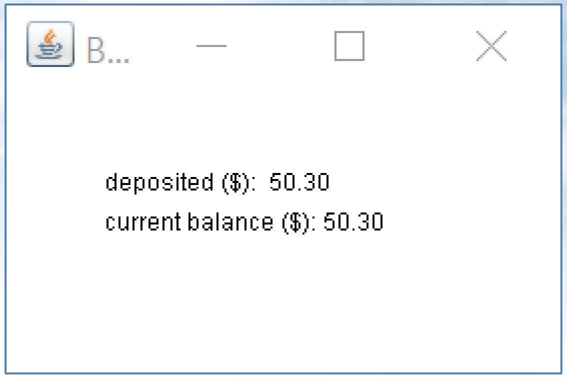
- Output
  - Current balance

07. Case Study: Bank Account Manager

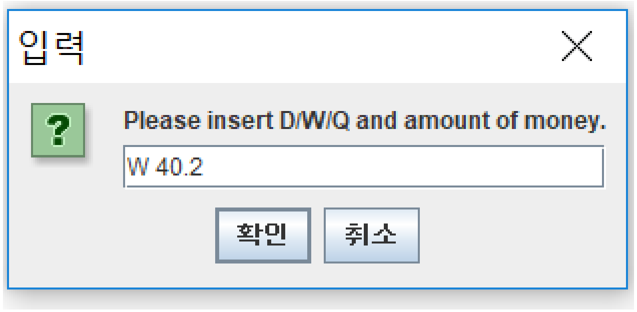
Usage



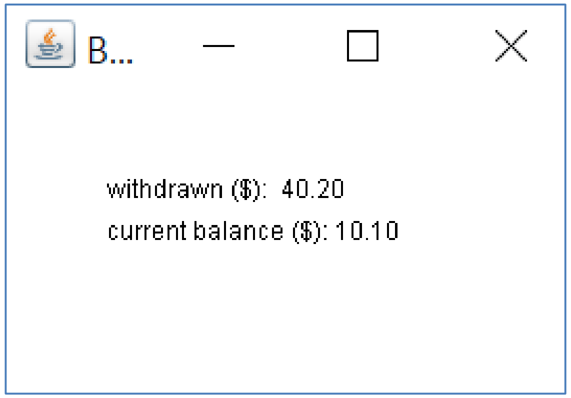
```
Please insert D/W/Q and amount of money.
D 50.3
```



```
deposited ($):  50.30
current balance ($): 50.30
```



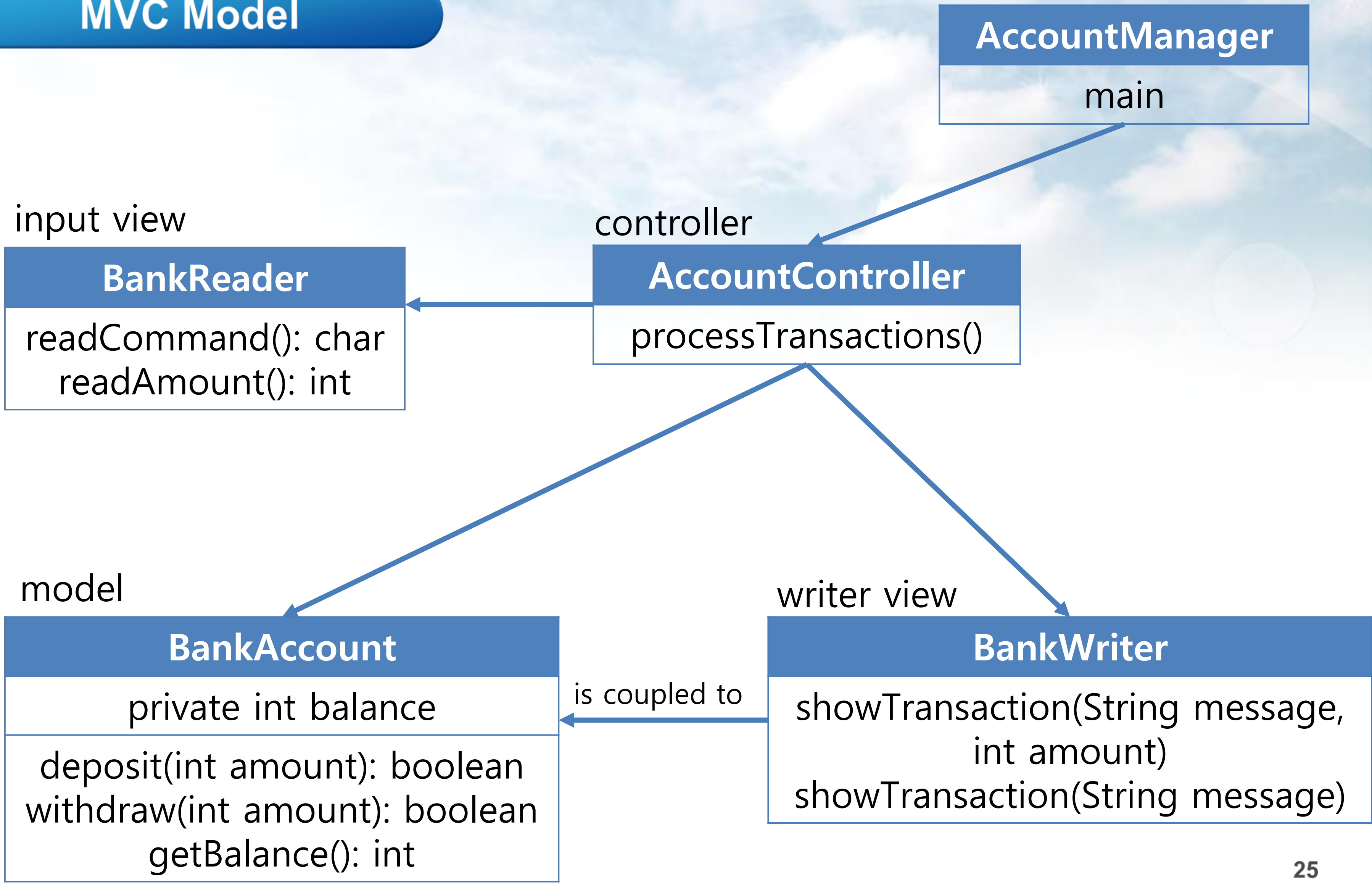
```
Please insert D/W/Q and amount of money.
W 40.2
```



```
withdrawn ($):  40.20
current balance ($): 10.10
```

07. Case Study: Bank Account Manager

MVC Model



07. Case Study: Bank Account Manager

BankAccount

class BankAccount	
constructor:	
BankAccount(int initial_balance)	Initialize the account
private attributes:	
Private int balance	Current balance
methods:	
getBalance(): int	Return current balance
deposit (int amount): boolean	Deposit amount from the account
withdraw (int amount): boolean	Withdraw amount from the account

07. Case Study: Bank Account Manager

BankAccount

```java
public class BankAccount
{
    private int balance;
    public BankAccount(int initial_amount)
    {
        if (initial_amount >= 0)
            balance = initial_amount;
        else
            balance = 0;
    }

    public int getBalance()
    {
        return balance;
    }
```

07. Case Study: Bank Account Manager

HANYANG UNIVERSITY

BankAccount.deposit

```
public boolean deposit(int amount)
{
    boolean result = false;
    if (amount < 0)
        JOptionPane.showMessageDialog(null, "invalid input.");
    else
    {
        balance = balance + amount;
        result = true;
    }
    return result;
}
```

07. Case Study: Bank Account Manager

BankAccount.withdraw

```
public boolean withdraw(int amount)
{
    boolean result = false;
    if (amount < 0)
        JOptionPane.showMessageDialog(null, "invalid input.");
    else if (amount > balance)
        JOptionPane.showMessageDialog(null, "not enough balance.");
    else
    {
        balance = balance - amount;
        result = true;
    }
    return result;
}
```

07. Case Study: Bank Account Manager

HANYANG UNIVERSITY

BankReader

class BankReader	
private attributes:	
Private String input_line	Recent command
methods:	
readCommand (String message): char	Return the first character of message
readAmount(): int	Return cents from the String xx.yy

07. Case Study: Bank Account Manager

BankReader.readCommand

```
public class BankReader
{

    private String input_line = "";

    public char readCommand(String message)
    {
        input_line =
        JOptionPane.showInputDialog(message).toUpperCase();
        return input_line.charAt(0);
    }
```

07. Case Study: Bank Account Manager

HANYANG UNIVERSITY

BankReader.readAmount

```
public int readAmount()
{
    int answer = 0;
    String s = input_line.substring(1, input_line.length());
    if(s.length() > 0)
    {
        double dollars_cents = new Double(s).doubleValue();
        answer = (int)(dollars_cents*100);
    }
    else
        JOptionPane.showMessageDialog(null, "input for amount is not
    provided.");
    return answer;
}
```

07. Case Study: Bank Account Manager

BankWriter

class BankWriter	
constructor:	
BankWriter(String title, BankAccount b)	Create a window
private attributes:	
Private int WIDTH, HEIGHT	Size of the window
Private BankAccount bank	Bank account
Private String last_transaction	Recent transaction message
methods:	
showTransaction (String message, int amount)	Print recent transaction
showTransaction (String message)	Print recent transaction

07. Case Study: Bank Account Manager

BankWriter

```
import java.awt.*; import javax.swing.*; import java.text.*;

public class BankWriter extends JPanel
{
    private int WIDTH = 300;
    private int HEIGHT = 200;
    private BankAccount bank;
    private String last_transaction = "";

    public BankWriter(String title, BankAccount b)
    {
        bank = b;
        JFrame f = new JFrame();
        f.getContentPane().add(this);
        f.setTitle(title);
        f.setSize(WIDTH, HEIGHT);
        f.setBackground(Color.white);
        f.setVisible(true);
    }
```

07. Case Study: Bank Account Manager

HANYANG UNIVERSITY

BankWriter.showTransaction

```
private String unconvert(int i)
{
       return new DecimalFormat("0.00").format(i/100.0);
}

public void showTransaction(String message, int amount)
{
    last_transaction = message + " " + unconvert(amount);
    this.repaint();
}

public void showTransaction(String message)
{
    last_transaction = message;
    this.repaint();
}
```

07. Case Study: Bank Account Manager

HANYANG UNIVERSITY

BankWriter.paintComponent

```
public void paintComponent(Graphics g)
{
    g.setColor(Color.white);
    g.fillRect(0, 0, WIDTH, HEIGHT);
    g.setColor(Color.black);
    int text_margin = 50;
    int text_baseline = 50;
    g.drawString(last_transaction, text_margin, text_baseline);
    g.drawString("current balance ($): " +
    unconvert(bank.getBalance()), text_margin, text_baseline + 20);
}
```

07. Case Study: Bank Account Manager

AccountController

class AccountController	
constructor:	
AccountController (BankReader r, BankWriter w, BankAccount b)	Initialize with objects
private attributes:	
Private BankReader reader	Input view
Private BankWriter writer	Output view
Private BankAccount account	Model: bank account
methods:	
processTransactoins()	Perform a transaction

07. Case Study: Bank Account Manager

AccountController

```
public class AccountController
{
    private BankReader reader;
    private BankWriter writer;
    private BankAccount account;

    public AccountController(BankReader r, BankWriter w, BankAccount a)
    {
        reader = r;
        account = a;
        writer = w;
    }
```

07. Case Study: Bank Account Manager

HANYANG UNIVERSITY

AccountController

```
public void processTransactions()
{
        char command = reader.readCommand("Please insert D/W/Q and amount of money.");
        switch (command)
        {
                case 'Q':
                                return;
                case 'D':
                {
                        int amount = reader.readAmount();
                        if (account.deposit(amount))
                                writer.showTransaction("deposited ($): ", amount);
                        else
                                writer.showTransaction("deposit error:  ", amount);
                        break;
                }
                case 'W':
                {
                        int amount = reader.readAmount();
                        if (account.withdraw(amount))
                                writer.showTransaction("withdrawn ($): ", amount);
                        else
                                writer.showTransaction("withdraw error: ", amount);
                        break;
                }
                default:
                                writer.showTransaction("invalid input: " + command);
        }
        this.processTransactions();
}
```

07. Case Study: Bank Account Manager

AccountManager

```
public class AccountManager
{
    public static void main(String[] args)
    {
        BankReader reader = new BankReader();
        BankAccount account = new BankAccount(0);
        BankWriter writer = new BankWriter("Bank account manager",
        account);
        AccountController controller = new AccountController(reader,
        writer, account);
        controller.processTransactions();
    }
}
```

08. Summary

HANYANG UNIVERSITY

Summary

- Conditional statements
  - if (question?) {statement1} else {statement2}
- Changing the flow
  - By exception, return, or System.exit
- Model-View-Controller model

Thanks

Week 6: Control Structure: Conditional Statements
Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)