

DECISIONS

Chapter 6

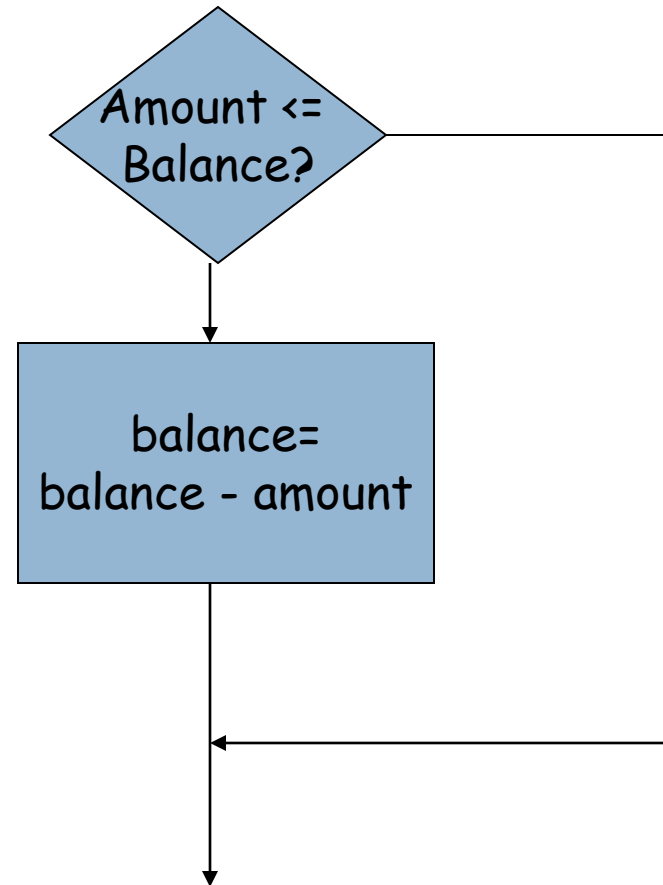
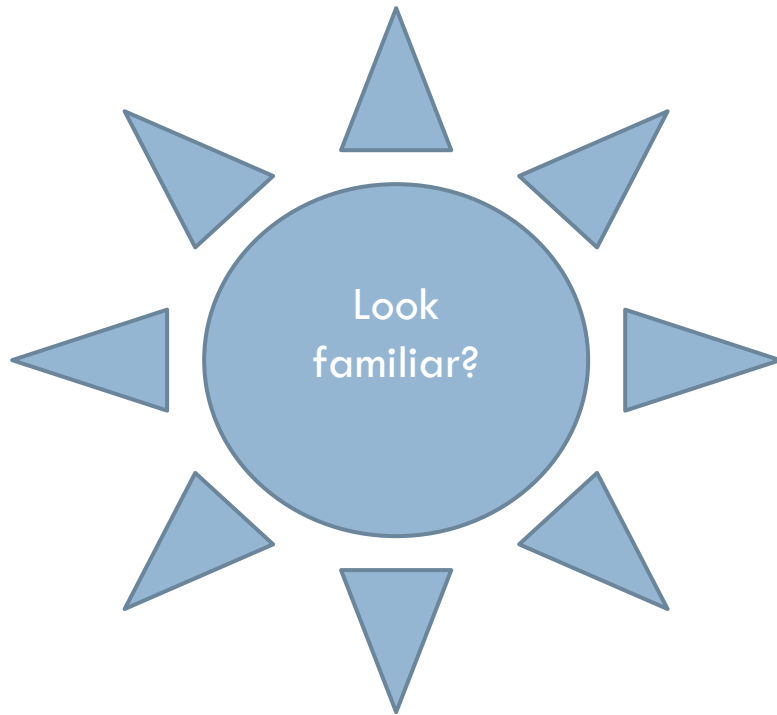
The if Statement

- Action based on a conditions
- If the condition is true, the body of the statement is executed

if (amount \leq balance)

 balance = balance – amount;

Flow Chart for If Statement

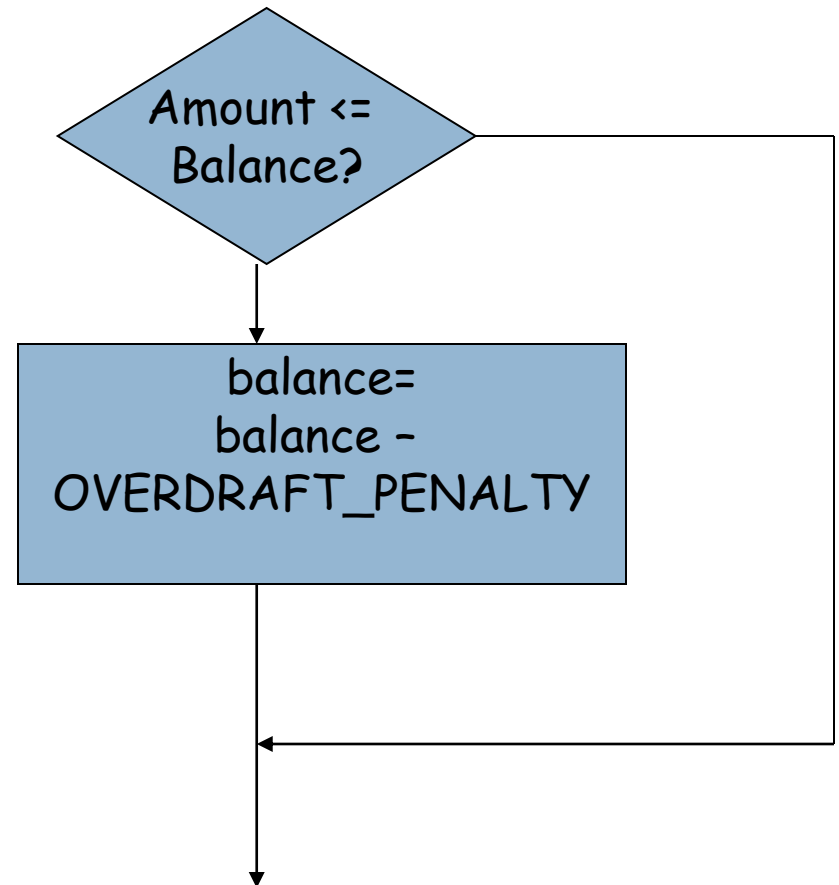


The if Statement (Java Form)

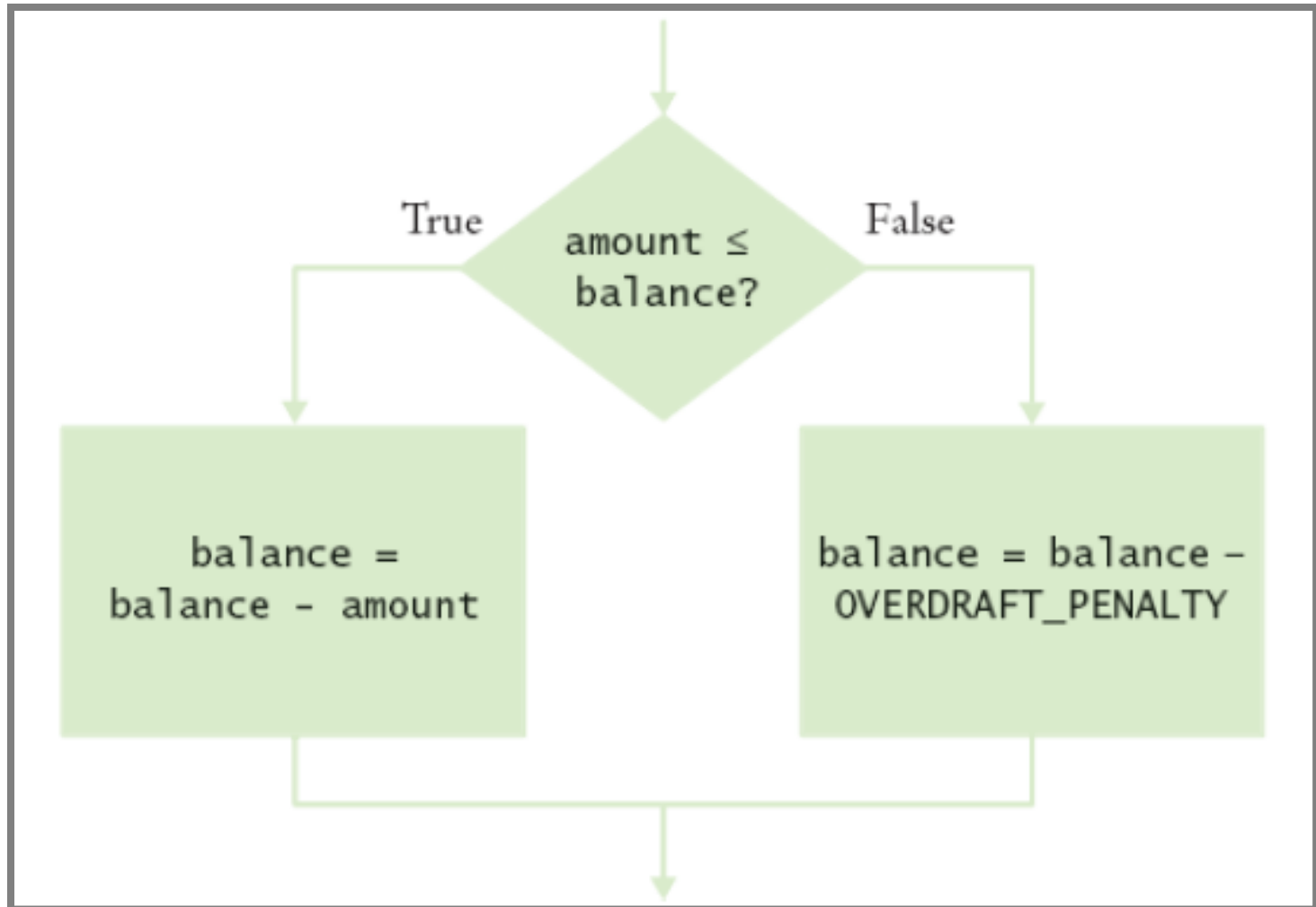
Java Statements_____.

```
If (amount <= balance)
    balance = balance - amount;
else
    balance = balance -
OVERDRAFT_PENALTY;
```

Visual Logic_____.



If Else Flowchart



Java Code for If Else

```
If (amount <= balance)
    balance = balance - amount;
else
    balance = balance -
        OVERDRAFT_PENALTY;
```

The if Statement

- What if you need to execute multiple statements based on decision?

```
if (amount <= balance)
{
    double balance = balance - amount;
}
```

What Is Wrong

```
if (amount <= balance)
    newBalance = balance - amount;
balance = newBalance;
```

```
if (amount <= balance)
{
    newBalance = balance - amount;
    balance = newBalance;
}
```


Relational Operators

- Test the relationship between two values.
- Most of the operators are similar to Visual Logic
- There are some differences. A big one is equals.
- In Visual Logic equals is `=`. In Java `==`.
- Not equal in Java is `!=`.

Relational Operators

Java	Description
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
==	Equal
!=	Not equal

Note: Equal and Not equal are different in Java than Visual Logic!

Comparing Floating-Point Numbers

- Must be careful in comparing
- Rounding problems



Java Example of Floating Point Comparison

Comparing floating – Point Numbers

- Need to test for close enough

```
final double EPSILON = 1E-14;
```

```
if (Math.abs(x-y) <= EPSILON)
```

```
    // x is approximately equal to y
```

Comparing Strings

- ❑ Can't use mathematical operators
- ❑ Special methods have been developed

`if (string1.equals(string2))....`

- ❑ Remember case matters
- ❑ Special method
`if (string1.equalsIgnoreCase(string2))....`

This is very different from Visual Logic

Comparing Strings

- May want to compare other than equal
`string1.compareTo(string2) < 0;`
Compares based on dictionary
- What happen if we use `==`
 - ▣ We are testing the string on left with constant
If (`nickname == "Rob"`)
 - ▣ Testing to see if the variable `nickname` value is `Rob`. Will be true only if it points to the string object

Remember that a string is and object and what is stored is the address. That is the problem

Comparing Strings

- ❑ `String nickname = "Rob";`
- ❑ `If(nickname == "Rob"); //Test is true`

- ❑ `String name="Robert";`
- ❑ `String nickname = name.substring(0,3);`
- ❑ `If(nickname == "Rob"); //Test is false`

Comparing Objects

- Use `==` you are testing to determine if the reference of the two objects are the same.
- In other words, do they point to the same address

```
Rectangle box1 = new Rectangle(5, 10, 20, 30);
```

```
Rectangle box2 = box1;
```

```
Rectangle box3 = new Rectangle(5, 10, 20, 30);
```

```
box1 == box2; // true or false
```

```
box1 == box 3; //true or false
```

Testing for Null

- ❑ Null is a special object reference
- ❑ It says no value has been set.
- ❑ The memory has been set aside or instantiated
- ❑ Null is not the same as the empty string ""
- ❑ "" assigns a blank
- ❑ Null means no assignment has been made
if (x = null).....

Multiple Alternatives

- Require more than 1 if/else decisions
- Need to make series of related comparisons
- Use
 - if
 - else if

Example

```
public String getDescription()  
{  
    if (richter >= 8.0)  
        r = "Most structures fall";  
    else if (richter >= 7.0)  
        r = "Many buildings destroyed";  
    .....  
    else  
        r= "Negative numbers are not valid";  
    return r;  
}
```

Switch Statement

- Use instead of a sequence of if/else/else statements.

```
int digit;
```

```
.....
```

```
Switch (digit)
```

```
{
```

```
    case 1: System.out.print("one");
```

```
    case 2: System.out.print("two");
```

```
    default: System.out.print("error");
```

```
}
```

Nested Branches

- Based on the decision of one statement make another decision.



Nested Branch in Visual Logic

Nested Branches - Java

```
If (status == SINGLE)
{
    if (income <= SINGLE_BRACKET1)
        tax = Rate1 * income;
    else if (income <= SINGLE_BRACKET2)
        tax = Rate2 * income.
```


Enumeration Types

- Think of it as a switch statement for strings.
- You assure that they are categorized correctly

```
public enum FilingStatus {SINGLE, MARRIED}
FilingStatus status = FilingStatus.SINGLE;
```
- Use the `==` to compare enumerated values

```
if (status==FilingStatus.SINGLE)...
```

Using Boolean Expressions

- True/False
- Assigned to an expression such as
 <1000
 double amount = 0;
 System.out.println(<1000);
Output: true

Predicate Methods

- Method that returns a boolean value

```
public class BankAccount
{
    public boolean isOverdrawn()
    {
        return balance < 0;
    }
}
```

Character Class Predicate Methods

- isDigit
- isLetter
- isUpperCase
- isLowerCase
- Scanner class
 - ▣ hasNext()
 - ▣ nextInt

Boolean Operators

□ Test two conditions

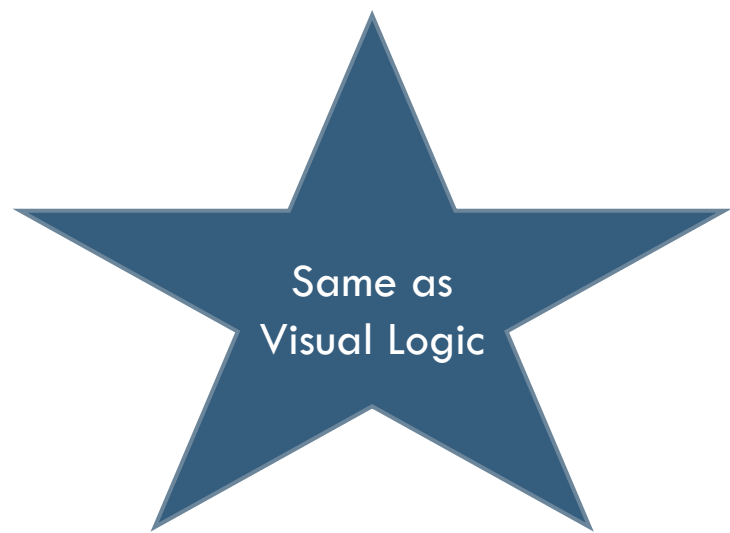
- ▣ and &&
- ▣ or ||

if (input.equals("S") || input.equals("M")).....
If(input.equals("S") && input.equals("M"))

- ▣ Can use ! – means not
- ▣ Negates the answer



Note
repeat



Same as
Visual Logic

Boolean Variables

- Primitive type
- Must declare as boolean

```
private boolean married;  
if (married)  
    ....
```
- Don't use

```
if (married == true)...
```