
Chapter 3

Decision Making

What is in This Chapter ?

In this chapter we discuss how to write code that makes decisions using the **if** statement and the **switch** statement.



3.1 Using the IF Statement

It is often necessary to be able to make a decision in a program and to act accordingly. For example, assume that we would like to examine a student's final mark in this course and print out an appropriate message. For example:

If the student's grade is less than 50 **then** print out "Oh well, there's always next term", **otherwise** print out "Congratulations!".

You will notice that this sentence is logically made up of 3 parts: (1) the part that checks whether or not the student passed (known as the "**condition statement**") (2) the part that specifies what to do if they do not pass (known as the "**if body**"), and (3) the part that specifies what to do if the student does pass (known as the "**else body**").

We can use the **if** statement in JAVA to make a similar decision like this:

```
if (grade < 50)
    System.out.println("Oh well, there's always next term.");
else
    System.out.println("Congratulations!");
```

Depending on the value of the **grade** variable, the code will print out only one of the two statements, but not both. Do you know why the following code works the same way ?

```
if (grade >= 50)
    System.out.println("Congratulations!");
else
    System.out.println("Oh well, there's always next term.");
```

In some cases, when making a decision, we do not need to "do something" in both cases. For example, what if we just wanted to print "Congratulations!" for students who passed, and do nothing in the case that they failed ? In this case, we could leave off the **else** part:


```
if (grade >= 50)
    System.out.println("Congratulations!");
```

In the examples above, we had one line of code to evaluate for each branch of the "**if**" condition. That is, we only had one statement to print each time. In general, when using an "**if**" statement, you are allowed to have multiple lines of JAVA code evaluated for each case. When you have more than one line, you need to insert some braces **{ }** after the **if** and **else** as follows ...

```
if (grade >= 50) {
    System.out.print("Congratulations! ");
    System.out.print(grade);
    System.out.println(" is a passing grade.");
}
else {
    System.out.print(grade);
    System.out.println(" is quite low. Oh well, there's always next term.");
}
```


It is often a good idea to use the braces anyway, even if you have only one line of code because it may prevent you from making some mistakes. For example, the following code is not the same as above:

```
if (grade >= 50)
    System.out.print("Congratulations! ");
    System.out.print(grade);
    System.out.println(" is a passing grade.");
else
    System.out.print(grade);
    System.out.println(" is quite low. Oh well, there's always next term.");
```




The code above will not compile. Since the brackets are missing, JAVA interprets the code as if there is only one line in the “if” body as follows:

```
if (grade >= 50)
    System.out.print("Congratulations! ");
System.out.print(grade);
System.out.println(" is a passing grade.");
else
    System.out.print(grade);
System.out.println(" is quite low. Oh well, there's always next term.");
```



It then sees the **else** as being out of place ... and will give a compile error saying: **‘else’ without ‘if’**. An even worse scenario is when JAVA does not notice the error at all. Consider the following:

```
if (grade >= 50)
    System.out.print("Congratulations! ");
    System.out.print(grade);
    System.out.println(" is a passing grade.");
System.out.println("All Done.");
```




In the above code, a **grade** of 75 will output the following:

```
Congratulations! 75 is a passing grade.  
All Done.
```

... and a grade of 25 will output this:

```
25 is a passing grade.  
All Done.
```

Clearly this is wrong. Also, be careful not to place a semi-colon **;** after the **if** statement brackets:

```
if (grade >= 50) ;  
    System.out.println("Congrats! " + grade + " is a passing grade.");
```

In the above code, a **grade** of 25 will output the following:

```
Congrats! 25 is a passing grade.
```

Why ? Because the semi-colon **;** at the end of the first line tells JAVA that there is no body for the **if** statement. Thus, the **System.out.println(...)** line is outside the **if** statement altogether and is therefore always evaluated.

Notice that we used **<** and **>=** in our examples above. These are called **logical operators** because they take two values, compare them, and then determine a logical **boolean** result of **true** or **false**. They are often used to compare numbers. Here is the list of logical operators that we can use:

- **<** less than
- **<=** less than or equal to
- **=** equal to
- **!=** not equal to
- **>=** greater than or equal to
- **>** greater than

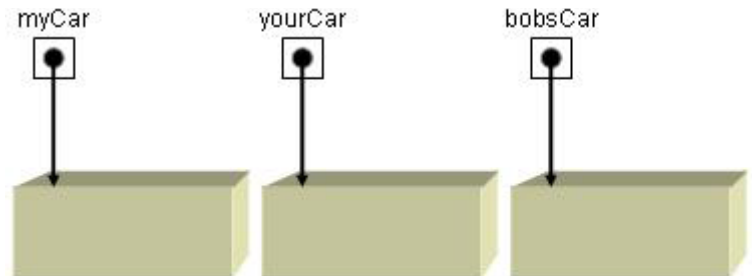
When writing an **if** statement, you must make sure that whatever is placed between the round brackets **()** is a JAVA expression that results in a **boolean** (i.e., the answer is either **true** or **false**).

It is necessary to examine the `==` operator for a moment. When dealing with primitive types, this operator basically tests whether or not the two primitives have the same value. So, `x == 5` will return **true** if `x` is equal to **5** (i.e., `x` has the value of **5**).

However, when dealing with objects, the `==` operator does not check to see if the objects are **equal**, instead it checks to see if the objects are **identical**. Two objects are only considered to be **identical** if they are the exact same object in the computer's memory.

Recall this code:

```
Car myCar, yourCar, bobsCar;
myCar = new Car();
yourCar = new Car();
bobsCar = new Car();
```

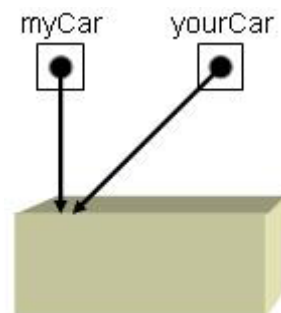


In this case, each **Car** is its own unique object in memory. Hence the following code would always produce **false**:

```
if (myCar == yourCar)
    System.out.println(true);
else
    System.out.println(false);
```

Consider, however, the following code:

```
Car myCar, yourCar;
myCar = new Car();
yourCar = myCar;
if (myCar == yourCar)
    System.out.println(true);
else
    System.out.println(false);
```



Now the code would always produce **true** because the object stored in the **myCar** variable is the exact same object stored in the **yourCar** variable.

If we really wanted to check if two objects were **equal** (i.e., they have the same values inside them), then we must use a special function called **equals()**. The **equals()** function is used as follows ...

```

Car    myCar, yourCar;

myCar = new Car();
yourCar = new Car();
if (myCar.equals(yourCar))
    System.out.println(true);
else
    System.out.println(false);

```

In this case, JAVA will check to determine whether or not the two Car objects have the same internal attributes (e.g., make, model, color ... whatever we defined as attributes).

Hence, the rule of thumb is that whenever you want to know if two **primitives** are equal, you use the **==** operator and whenever you want to know if two **objects** are equal, you use the **equals()** function.



But, there is a word of caution. Most of the existing pre-defined objects in JAVA (e.g., String, Date, etc..) have an appropriate **equals()** function that properly checks equality. However, when you create your own objects (e.g., **Person**, **Car**, **BankAccount**), the **equals()** function will by default still only check whether the objects are *identical*. Hence, you would need to write your own **equals()** method to make sure that it properly checks for equality. We will discuss this in a later chapter.

Consider now an example in which we take the number grade of the student (i.e., from 0% to 100%) and output a letter grade (from F to A+). How would we do this? Well ... we would need to understand which letter grade corresponds to which number grades:

A = 80% - 100%	D = 50% - 59%
B = 70% - 79%	F = 0% - 49%
C = 60% - 69%	

So, given a grade, how can we output the appropriate letter? We could use the **if** statement:

```

if (grade >= 80)
    System.out.println("A");
else {
    if (grade >=70)
        System.out.println("B");
    else {
        if (grade >= 60)
            System.out.println("C");
        else {
            if (grade >= 50)
                System.out.println("D");
            else
                System.out.println("F");
        }
    }
}

```

You may notice now that we have an **if** statement inside of an **else** statement's body. This is known as **nested if** statements. Notice how the code is indented carefully so that when reading the code we can see what is inside each **if/else** statement's body.

In fact, an **if/else** statement is actually considered a single JAVA statement. So, we do not need the braces here. In fact, we can even place the succeeding **if** statements up on the same line as the **else** statements and align everything up on the left. The following is how we usually write such code:

```

if (grade >= 80)
    System.out.println("A");
else if (grade >=70)
    System.out.println("B");
else if (grade >= 60)
    System.out.println("C");
else if (grade >= 50)
    System.out.println("D");
else
    System.out.println("F");

```

Consider another example in which we are given an integer representing a **month** and we would like to store (in a variable called **days**) the number of days in that month (we will assume that it is not a leap year). Here is the table of information that we need to know:

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Days	31	28	31	30	31	30	31	31	30	31	30	31

Here is how we can do this with **if** statements:

```

int month, days;

month = ... // assume that we got this from the user

if (month == 1)
    days = 31;
else if (month == 2)
    days = 28;
else if (month == 3)
    days = 31;
else if (month == 4)
    days = 30;
... etc ...

```

However, you can see that the **if** statement will be 24 lines long! Since there are only 3 values for the months (i.e., 31, 30 and 28), there should be a way to arrange it in a format like this ...

```

int month, days;

month = ...    // assume that we got this from the user
if (...)      // Jan, Mar, May, Jul, Aug, Oct, Dec
    days = 31;
else if (...)  // Apr, Jun, Sep, Nov
    days = 30;
else if (...)  // Feb only
    days = 28;

```

To do this, we need to have some way of asking whether or not the month is Jan **or** Mar **or** May **or** Jul etc... Well, it is a good thing then that JAVA supplies us with what is known as **Boolean operators**. Here are three useful **boolean** operators:

- **&&** conditional **and**
- **||** conditional **or**
- **!** **not** (prefix)

The **&&** and **||** operators are used in-between two JAVA expressions that evaluate to booleans. Below is a table explaining the results of using these two boolean values **b1** and **b2** in various expressions:

b1	b2	if (b1 && b2)	if (b1 b2)	if (!b1)	if (b1)
false	false	false	false	true	false
false	true	false	true	true	false
true	false	false	true	false	true
true	true	true	true	false	true

Notice that the **&&** results in **true** only when both booleans are **true**, and **false** otherwise. Conversely, the **||** results in **false** only when both booleans are **false**, and **true** otherwise. Also note that the **!** results in the opposite value of the boolean.

We can actually combine multiple booleans operators together in various ways:

```

if ((boolean1 && boolean2 && boolean3) || (boolean1 && !boolean2)) ...

```

So how do we use this in our month/days example ? Well, each boolean expression can be in a format something like this: **(month == 2)**. Therefore, our solution may look like this:

```

if ((month == 1) || (month == 3) || (month == 5) || (month == 7) ||
    (month == 8) || (month == 10) || (month == 12))
    days = 31;
else if ((month == 4) || (month == 6) || (month == 9) || (month == 11))
    days = 30;
else if (month == 2)
    days = 28;

```


So we just used a bunch of “or” operators. We can actually simplify the code by noticing something interesting. Since the first two **if** statements checked 11 of the 12 months, then we do not need to ask **if (month == 2)** in the last if statement because it is the only possible month remaining (assuming that the month was in the valid range of 1 to 12). Also, it does not matter which order we check the months in. So, the following code will also do the same thing, but is much shorter:


```
if (month == 2)
    days = 28;
else if ((month == 4) || (month == 6) || (month == 9) || (month == 11))
    days = 30;
else
    days = 31;
```

As one more example, how would we use an **if** statement that prints out the message “valid” when a given **number** is within a given range (e.g., from 1 to 10) and “invalid” otherwise? Here is a common mistake that many students make:

```
int number;

number = ... ; //code left out intentionally

if (1 <= number <= 10)
    System.out.println("Valid");
else
    System.out.println("Invalid");
```



JAVA does not allow us to check ranges in this manner. Instead, we have to check the two sides of the range separately.

```
int number;

number = ... ; //code left out intentionally

if ((number >= 1) && (number <= 10))
    System.out.println("Valid");
else
    System.out.println("Invalid");
```

Consider this example that defines three **Person** objects and then tries to determine the oldest person. The code is not complete ... it uses pseudo-code. How would you complete it? ...

```

Person    p1, p2, p3;

p1 = new Person("Hank", "Urchif", 19, 'M', false);
p2 = new Person("Holly", "Day", 67, 'F', true);
p3 = new Person("Bobby", "Socks", 12, 'M', false);

// Write code to set the oldest variable to be the oldest person.
// Remember that we can say p.age to get the age of person p
//
// Person oldest;
// IF (the 1st person is older than the 2nd and 3rd person)
//   then the oldest should be the 1st person;
// OTHERWISE IF (the 2nd person is older than the 1st and 3rd person)
//   then the oldest should be the 2nd person;
// OTHERWISE
//   the oldest should be the 3rd person;

```

Here is one solution. Do you see how it follows from the pseudo code ?

```

Person    p1, p2, p3;

p1 = new Person("Hank", "Urchif", 19, 'M', false);
p2 = new Person("Holly", "Day", 67, 'F', true);
p3 = new Person("Bobby", "Socks", 12, 'M', false);

// Write code to set the oldest variable to be the oldest person.
Person    oldest;

if ((p1.age > p2.age) && (p1.age > p3.age))
    oldest = p1;
else if ((p2.age > p1.age) && (p2.age > p3.age))
    oldest = p2;
else
    oldest = p3; //Assumes that there are no equal ages

```

Now what code would you write if you wanted print out a message saying “All retired” if all three people are retired. If none are retired then print out “None retired”. Here is the “pseudo” code:

```

IF (the 1st, 2nd and 3rd person are all retired) then print “All Retired”;
IF (the 1st, 2nd and 3rd person are all not retired) then print “None Retired”;

```

Here is the solution in actual JAVA code ...

```
if (p1.retired && p2.retired && p3.retired)
    System.out.println("All Retired");
if (!p1.retired && !p2.retired && !p3.retired)
    System.out.println("None Retired");
```

Lastly, assume that we wanted to have a special “Grandma/Granddaughter Night” at the theatre and wanted to give a discount of 50% to women that are retired or to girls who are 12 and under. The discount should otherwise be 0%. How would we use **if** statements to examine a **Person p** and appropriately set the **discount** variable ?

```
Person    p = new Person();

... // some code omitted to set the name, age, gender, etc...

// Write code to compute the appropriate discount
int discount;

IF (the person is female and if they are under 13 or retired)
    then the discount is 50%
OTHERWISE
    the discount is 0%
```

Here is the solution in actual JAVA code:

```
Person    p = new Person();

... // some code omitted to set the name, age, gender, etc...

// Write code to compute the appropriate discount
int discount = 0;

if ((p.gender == 'F') && (p.age < 13 || p.retired))
    discount = 50;
```

As you can see, working with the **IF** statements in JAVA is quite easy as long as you know what objects you have and what information is inside of them.

Supplemental Information: The Selection Operator

In addition to the **if** statement, JAVA allows a **Selection Operator** to be used. Here is the general format:

```
<booleanCondition> ? <valueIfTrue> : <valueIfFalse>
```

The result of the expression is one of the two values supplied, depending on whether the condition is **true** or **false**. For example,

```
String result;  
result = grade > 50 ? "pass" : "fail";
```

does the same thing as:

```
String result;  
  
if (grade > 50)  
    result = "pass";  
else  
    result = "fail";
```

So this saves the amount of code to write, but it does sacrifice a little bit of *readability* of the code.

3.2 The Switch Statement

In addition to the **if** statement, there is another construct in JAVA called the **switch** statement which is beneficial for simplifying code that contains *nested if* statements.

Consider a simplified letter grade that is given for a course project (i.e., **A, B, C, D, F**). (i.e., for the purpose of brevity, we will assume that there is no **A+, A-, B+, B-** ... grades). Sometimes when a student receives a letter grade he/she would like to know what percentage range corresponds to that letter. For example, a **B** corresponds to a grade between 70% and 79%.

Consider code that uses **if** statements to compute the proper range as follows ...

```

char    aLetter;

aLetter = ...; // the code for obtaining the grade has been omitted

if (aLetter == 'A')
    System.out.println("80% - 100%");
else if (aLetter == 'B')
    System.out.println("70% - 79%");
else if (aLetter == 'C')
    System.out.println("60% - 69%");
else if (aLetter == 'D')
    System.out.println("50% - 59%");
else if (aLetter == 'F')
    System.out.println("0% - 49%");
else
    System.out.println("not defined");

```

Looking at the code, we can see that there are 5 **if** statements and it looks very messy. If we later decide to handle the A+, A-, B+, etc.. cases, then the code will look much longer and cluttered. There is a better way to write this code. We can use a **switch** statement. The **switch** statement is typically used in situations where we have a sequence of nested **if** statements in which only one of the **if** statements is to be executed.



The switch statement has the following format in JAVA:

```

switch (aPrimitiveExpression) {
    case val1:
        /*one or more lines of JAVA code*/;
        break;
    case val2:
        /*one or more lines of JAVA code*/;
        break;
    ...
    case valN:
        /*one or more lines of JAVA code*/;
        break;
    default:
        /*one or more lines of JAVA code*/;
        break;
}

```

In the above code, **aPrimitiveExpression** is either a primitive variable (e.g., a variable of type **int**, **char**, **float**, etc...) or any JAVA code that results in a primitive value. The values of **val1**, **val2**, ..., **valN** must all be primitive constant values of the same type as **aPrimitiveExpression**.

The **switch** statement works as follows:

1. It evaluates **aPrimitiveExpression** to obtain a value (the expression MUST result in a primitive data type, it **cannot** be an object).
2. It then checks the values **val1**, **val2**, ..., **valN** in order from top to bottom until a value is found equal to the value of **aPrimitiveExpression**. If none match, then the **default** case is executed.
3. It then evaluates the statements corresponding to the **case** whose value matched.
4. If there is a **break** at the end of the lines of JAVA code for that **case**, then the **switch** statement quits. Otherwise it continues to evaluate all the successive **case** statements that follow ... until a **break** is found or until no more cases remain.

Here is how can we make use of the switch statement for solving the grade range problem ?

```
char    aLetter;

aLetter = ...; // the code for obtaining the grade has been omitted

switch(aLetter) {
    case 'A': System.out.println("80% - 100%"); break;
    case 'B': System.out.println("70% - 79%"); break;
    case 'C': System.out.println("60% - 69%"); break;
    case 'D': System.out.println("50% - 59%"); break;
    case 'F': System.out.println("0% - 49%"); break;
    default:  System.out.println("not defined");
}
```

We can clearly see that the code is simpler to read. However, this is not the only advantage of a **switch** statement. Consider our previous example in which we were given an integer representing a month and we would like to know the number of days in that month:

```
if (month == 2)
    days = 28;
else if ((month == 4) || (month == 6) || (month == 9) || (month == 11))
    days = 30;
else
    days = 31;
```

Here is how we can use a **switch** statement ...

```

switch(month) {
    case 2: days = 28; break;
    case 4:
    case 6:
    case 9:
    case 11: days = 30; break;
    default: days = 31;
}

```

Note that when the month is 4, 6, 9, or 11, then the **days = 30;** is evaluated. The code is not necessarily much shorter, but it is simpler to read. This is the main advantage of a **switch** statement.

One thing that needs mentioning is that the value of the cases must be **primitive literals**. That is, they cannot be expressions, ranges (nor Strings, since Strings are objects). Nor can we make use of the logical operators such as **and**'ing and **or**'ing.

So these two examples will not work:

```

switch (age) {
    case 1-12: price = 5.00; break; // Won't compile
    case 13-17: price = 8.00; break; // Won't compile
    case 18-54: price = 10.00; break; // Won't compile
    default: price = 6.00;
}

```



```

switch (name) {
    case "Mark": bonus = 3; break; // Won't compile
    case "Betty": bonus = 2; break; // Won't compile
    case "Jane": bonus = 1; break; // Won't compile
    default: bonus = 0;
}

```



3.3 A Decision Making Example Program

Consider writing a program that will be placed at a kiosk in front of a bank to allow customers to determine whether or not they qualify for the bank's new "Entrepreneur Startup Loan". Assume that this kind of loan is only given out to someone who is currently employed and who is a recent University graduate, or someone who is employed, over 30 and has at least 10 years of full-time work experience.

The program should display information to the screen as well as ask the user various questions ... and then determine if the person qualifies. Using the **Scanner** object to get user input, and various variables to store the input, lets try to write such a program.

Here is a start to the program that displays some opening instructions:

```
import java.util.Scanner;

class LoanQualificationProgram {
    public static void main(String args[]) {
        // Get a Scanner object for user input
        Scanner keyboard = new Scanner(System.in);

        // Display some opening instructions
        System.out.println("Bank of Java");
        System.out.println("=====");
        System.out.println("Follow the instructions below to " +
            "determine whether or not you qualify " +
            "for a Entrepreneur Startup Loan...\n");
    }
}
```

Now we need to start asking the user some questions. We can first ask whether or not he/she is employed. Likely we will ask for a character such as 'y', 'Y', 'n' or 'N'. We can then check the input and store the employment status as a **boolean**. Here is the new code to add:

```
char    charInput;
boolean employed;

// Determine whether or not the user is employed
System.out.print("Are you currently employed (Y/N)? ");
charInput = keyboard.nextLine().charAt(0);
if ((charInput == 'y') || (charInput == 'Y'))
    employed = true;
else
    employed = false;
```

We can similarly ask whether or not they have a recent University degree:

```
boolean    hasDegree;

// Determine if the user has a recent University degree
System.out.print("Did you graduate with a University degree " +
    "in the past 6 months (Y/N)? ");
charInput = keyboard.nextLine().charAt(0);
if ((charInput == 'y') || (charInput == 'Y'))
    hasDegree = true;
else
    hasDegree = false;
```


In a similar manner, we can ask for the user's age and the number of years that they have worked at full time status:

```
int         age, yearsWorked;

// Determine the user's age
System.out.print("How old are you ?  ");
age = keyboard.nextInt();

// Determine the number of years worked at full time status
System.out.print("How many years have you been working " +
                "at full time status ?  ");
yearsWorked = keyboard.nextInt();
System.out.println("\n");
```

Finally, we can determine whether or not they qualify:

```
char        charInput;
boolean     employed, hasDegree;
int         age, yearsWorked;

...

// Now determine whether or not the person qualifies for the loan
if (employed) {
    if (hasDegree)
        System.out.println("Congratulations! You qualify " +
                            "for the ESL loan.");
    else {
        if (age >= 30) {
            if (yearsWorked >= 10)
                System.out.println("Congratulations! You qualify " +
                                    "for the ESL loan.");
            else
                System.out.println("Sorry, you must have worked " +
                                    "at least 10 years at full " +
                                    "time status to qualify.");
        }
        else
            System.out.println("Sorry, you must be a recent " +
                                "University graduate or be at " +
                                "least 30 years of age.");
    }
}
else {
    System.out.println("Sorry, you must be currently " +
                        "employed to qualify.");
}
```

Here is the code altogether ...

```
import java.util.Scanner;

class LoanQualificationProgram {
    public static void main(String args[]) {

        char        charInput;
        boolean      employed, hasDegree;
        int          age, yearsWorked;

        // Get a Scanner object for user input
        Scanner keyboard = new Scanner(System.in);

        // Display some opening instructions
        System.out.println("Bank of Java");
        System.out.println("=====");
        System.out.println("Follow the instructions below to " +
            "determine whether or not you qualify " +
            "for a Entrepreneur Startup Loan...\n");

        // Determine whether or not the user is employed
        System.out.print("Are you currently employed (Y/N)? ");
        charInput = keyboard.nextLine().charAt(0);
        if ((charInput == 'y') || (charInput == 'Y'))
            employed = true;
        else
            employed = false;

        // Determine if the user has a recent University degree
        System.out.print("Did you graduate with a University degree " +
            "in the past 6 months (Y/N)? ");
        charInput = keyboard.nextLine().charAt(0);
        if ((charInput == 'y') || (charInput == 'Y'))
            hasDegree = true;
        else
            hasDegree = false;

        // Determine the user's age
        System.out.print("How old are you ? ");
        age = keyboard.nextInt();

        // Determine the number of years worked at full time status
        System.out.print("How many years have you been working " +
            "at full time status ? ");
        yearsWorked = keyboard.nextInt();
        System.out.println("\n");

        // Now determine whether or not the person qualifies for the loan
        if (employed) {
            if (hasDegree)
                System.out.println("Congratulations! You qualify " +
                    "for the ESL loan.");
        }
    }
}
```

```

        else {
            if (age >= 30) {
                if (yearsWorked >= 10)
                    System.out.println("Congratulations! You qualify " +
                                       "for the ESL loan.");
                else
                    System.out.println("Sorry, you must have worked " +
                                       "at least 10 years at full " +
                                       "time status to qualify.");
            }
            else
                System.out.println("Sorry, you must be a recent " +
                                   "University graduate or be at " +
                                   "least 30 years of age.");
        }
    }
    else
        System.out.println("Sorry, you must be currently " +
                            "employed to qualify.");
}
}
}

```

Here is the output from three sample runs of the program:

```

Bank of Java
=====
Follow the instructions below to determine whether or not you qualify for a
Entrepreneur Startup Loan...

Are you currently employed (Y/N)? Y
Did you graduate with a University degree in the past 6 months (Y/N)? N
How old are you ? 28
How many years have you been working at full time status ? 8

Sorry, you must be a recent University graduate or be at least 30 years of age.

```

```

Bank of Java
=====
Follow the instructions below to determine whether or not you qualify for a
Entrepreneur Startup Loan...

Are you currently employed (Y/N)? Y
Did you graduate with a University degree in the past 6 months (Y/N)? Y
How old are you ? 22
How many years have you been working at full time status ? 1

Congratulations! You qualify for the ESL loan.

```

Bank of Java

=====

Follow the instructions below to determine whether or not you qualify for a
Entrepreneur Startup Loan...

Are you currently employed (Y/N)? **N**

Did you graduate with a University degree in the past 6 months (Y/N)? **Y**

How old are you ? **24**

How many years have you been working at full time status ? **3**

Sorry, you must be currently employed to qualify.