ICS 141 Programming with Objects

Jessica Maistrovich

Metropolitan State University

Boolean Expressions

Conditions (or Boolean Expressions)

- A condition is basically a **Boolean expression** that evaluates to *true* or *false*.

Relational Operators	
Operator	**Meaning**
A == B	is A equal to B ?
A < B	is A less than B ?
A <= B	is A less than or equal to B ?
A > B	is A Greater than B ?
A >= B	is A Greater than or equal to B ?
A != B	is A not equal to B ?

Slide provided by Thanaa Ghanem

Logical Operators

- Logical Operators are used to create compound conditions

- Three are three logical operators:
  - **||** (logical OR)
    - A binary expression formed with || evaluates to true if any one of its components is true
  - **&&** (logical AND)
    - A binary expression formed with && evaluates to true if all of its components are true
  - **!** (logical NOT, logical negation)

Slide provided by Thanaa Ghanem

Example on using AND (&&)

- Assume a car rental agency wants a program to determine who can rent a car. The rules are:
  - A renter must be 21 years old or older.
  - A renter must have a credit card with $10,000 or more of credit.

```
if (age >= 21 && credit >= 10000){
   System.out.println("You can rent! ");
}else{
   System.out.println("You cannot rent! ");
}
```

Could a 30 year old with $10000 credit rent a car?

Slide provided by Thanaa Ghanem

Example on using OR (||)

- You would like to buy a $25,000 sports car. To pay for the car you need either enough cash or enough credit.

```
if (cash >= 25000 || credit >= 25000){
   System.out.println("Enough to buy the car! ");
}else{
   System.out.println("Sorry! You cannot buy the car");
}
```

Example on using Not (!)

- You are shopping for new shoes. You are only interested in shoes that cost less than $50. Here is how you program your decision.

```
if ( !(cost < 50) ){
        System.out.println("Reject these shoes");
}else{
        System.out.println("Acceptable shoes");
}
```

The following code is not correct because ! is applied to `cost`

```
if ( !cost < 50 ) {
        System.out.println("Reject these shoes");
}else{
        System.out.println("Acceptable shoes");
}
```

Slide provided by Thanaa Ghanem

CAUTION: Range testing

- The following expression is correct in Mathematics but **incorrect** in Java programming

```
1 <= numOfDays <= 31
```

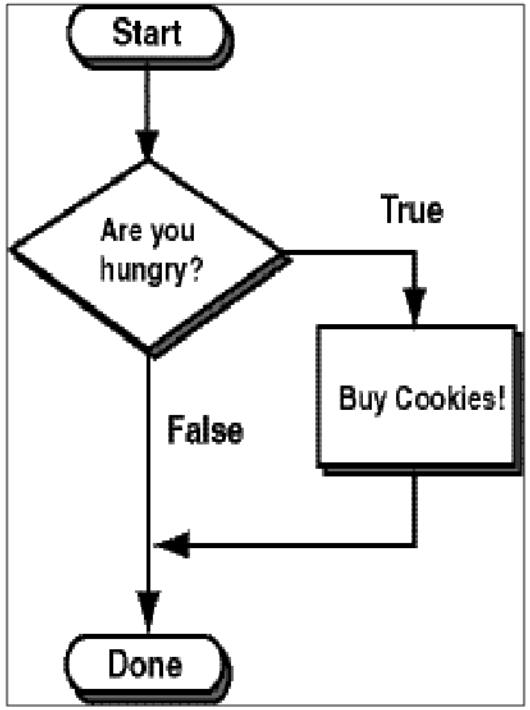
- The equivalent expression in programming is:

```
1 <= numOfDays && numOfDays <= 31
```

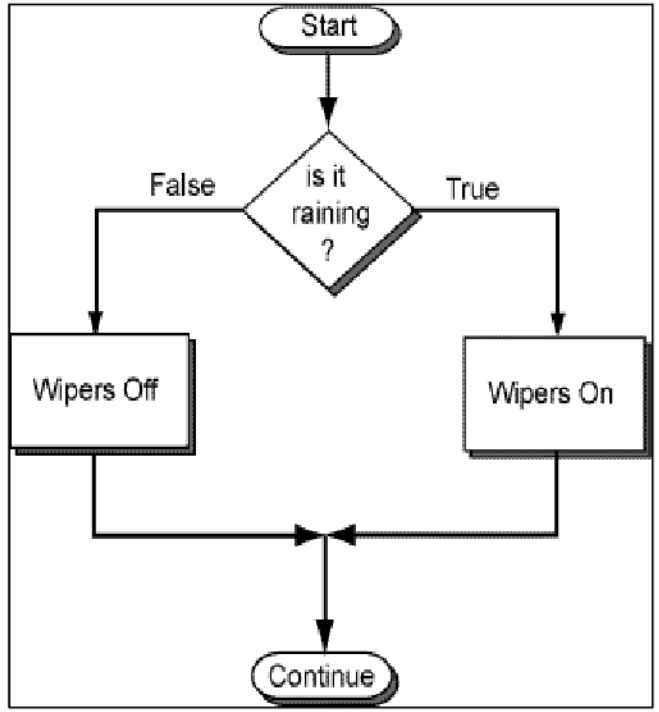
If

Types of decisions

Single-branch



Two-way



Slide provided by Thanaa Ghanem

if statement Syntax

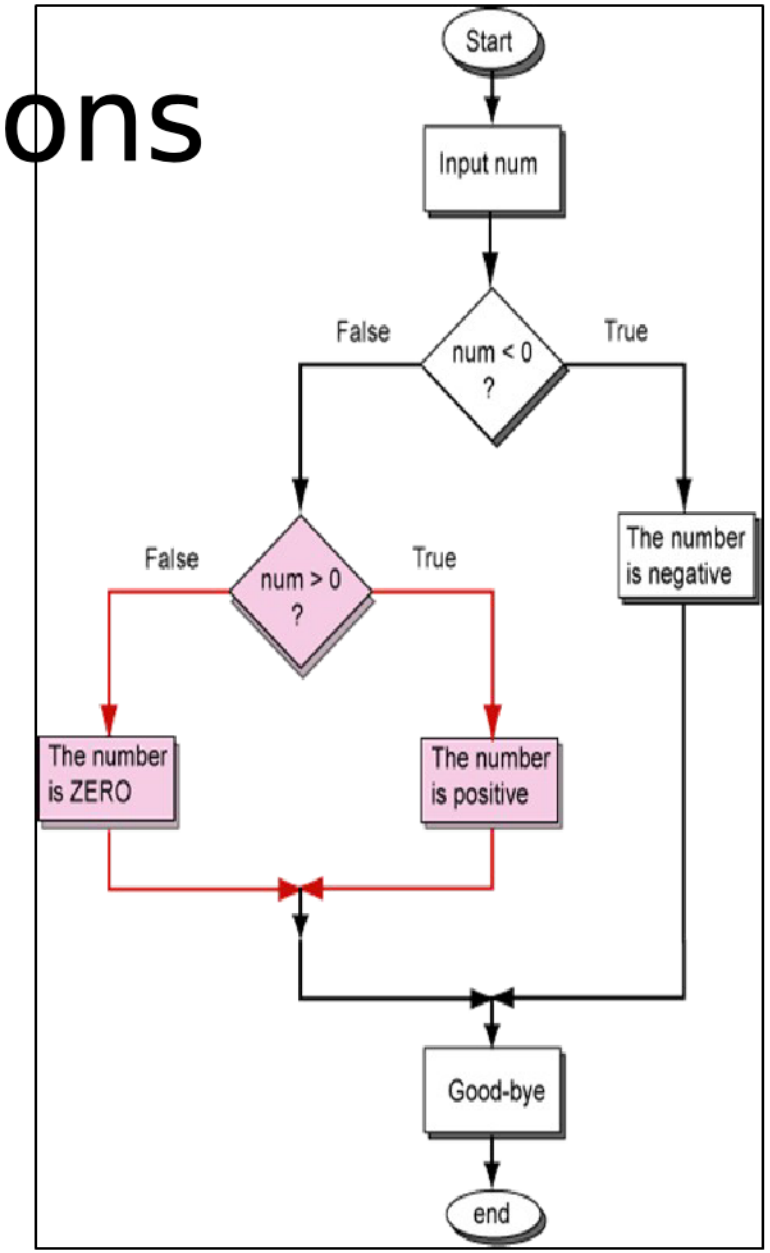
	Python	Java

Syntax	`if condition:`
	`  indented statement block-1`
	`else:`
	`  indented Statement block-2`
		`if (condition){`
		`  statement block-1`
		`}else{`
		`  statement block-2`
		`}`

Example	`if age < 13:`
	`  print ("Child")`
	`else:`
	`  print (”Not a child")`
		`if (age < 13){`
		`  System.out.println`
		`  ("Child");`
		`}else{`
		`  System.out.println`
		`  (" Not a child");`
		`}`

Slide provided by Thanaa Ghanem

Multiple Branch Decisions



Multi-branch `if` statement

Python

```python
if condition-1:
    indented statement block-1
elif condition-2:
    indented Statement block-2
elif condition-3:
    indented Statement block-3
elif condition-4:
    indented Statement block-4
else:
     indented Statement block-5
```

Java

```java
if (condition-1){
    statement block-1
}else if (condition-2) {
    statement block-2
}else if (condition-3) {
    statement block-3
}else if (condition-4) {
    statement block-4
}else {
    statement block-5
}
```

Slide provided by Thanaa Ghanem

Multi-branch `if` statement

```
if (condition-1){
    statement block-1
}else if (condition-2) {
    statement block-2
}else if (condition-3) {
    statement block-3
}else if (condition-4) {
    statement block-4
}else {
    statement block-5
}
```

- The conditions are checked in order
- Only one statement will be executed which corresponds to the to the first condition that is evaluated to true
- **Order matters!**

Slide provided by Thanaa Ghanem

Note

```java
if i > 0 {
  System.out.println("i is positive");
}
```

(a) Wrong

```java
if (i > 0) {
  System.out.println("i is positive");
}
```

(b) Correct

```java
if (i > 0) {
  System.out.println("i is positive");
}
```

(a)

Equivalent

```java
if (i > 0)
  System.out.println("i is positive");
```

(b)

Slide provided by Thanaa Ghanem

CAUTION

A `return` statement is required for a value-returning method. The method shown below in (a) is logically correct, but it has a compilation error because the Java compiler thinks it possible that this method does not return any value.

```java
public static int sign(int n) {
  if (n > 0)
    return 1;
  else if (n == 0)
    return 0;
  else if (n < 0)
    return -1;
}
```

(a)

Should be

```java
public static int sign(int n) {
  if (n > 0)
    return 1;
  else if (n == 0)
    return 0;
  else
    return -1;
}
```

(b)

To fix this problem, delete *if (n < 0)* in (a), so that the compiler will see a return statement to be reached regardless of how the `if` statement is evaluated.

Slide provided by Thanaa Ghanem

Try it!

- In the Account class, adjust the withdraw method so that the account balance does not go below zero. If there is enough money for the withdrawal, perform the action and return true. Otherwise, return false.

Switch

.Alternative to if/else structure if all conditions are using a comparison for one variable.

Revisit if/else if

Write a program that reads a one-digit number from the user then prints on the screen the digit in letters.

```
int digit;
System.out.println("Enter a single digit number:");
digit = input.nextInt();


if (digit == 0){
            System.out.println(" ZERO ");
}else if (digit==1){
            System.out.println(" ONE ");
}else if (digit==2){
            System.out.println(" TWO ");
            …and so on….
}else{
            System.out.println(" Not a single-digit number");
      }
```

Slide provided by Thanaa Ghanem

switch Statement Rules

The switch-expression must yield a value of char, byte, short, int, or String type and must always be enclosed in parentheses.

The value1, ..., and valueN must have the same data type as the value of the switch-expression and must be a constant variable or a literal value.

The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression.

```
switch (switch-expression) {
  case value1:
statement(s)1;
        break;
  case value2:
       statement(s)2;
        break;
  …
  case valueN:
       statement(s)N;
        break;
  default:
       statement(s)-for default;
}
```

`switch` Statement Rules (continued)

The keyword `break` is optional, but it should be used at the end of each case in order to terminate the remainder of the `switch` statement. If the `break` statement is not present, the next `case` statement will be executed.

```
switch (switch-expression) {
  case value1:          statement(s)1;
        break;
  case value2:          statement(s)2;
        break;
  …
  case valueN:          statement(s)N;
        break;
  default:
        statement(s)-for-default;
}
```

The `default` case, which is optional, can be used to perform actions when none of the specified cases matches the switch-expression.

The `case` statements are executed in sequential order, but the order of the cases (including the `default` case) does not matter. However, it is good programming style to follow the logical sequence of the cases and place the default case at the end.

Suppose ch is 'a':

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

ch is 'a':

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

Execute this line

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

Execute this line

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

Slide provided by Thanaa Ghanem

Execute this line

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

Slide provided by Thanaa Ghanem

Execute next statement

```
switch (ch) {
  case 'a': System.out.println(ch);
  case 'b': System.out.println(ch);
  case 'c': System.out.println(ch);
}
```

Next statement;

Suppose ch is 'a':

```
switch (ch) {
  case 'a': System.out.println(ch);
            break;
  case 'b': System.out.println(ch);
            break;
  case 'c': System.out.println(ch);
}
```

ch is 'a':

```
switch (ch) {
  case 'a': System.out.println(ch);
            break;
  case 'b': System.out.println(ch);
            break;
  case 'c': System.out.println(ch);
}
```

Execute this line

```
switch (ch) {
  case 'a': System.out.println(ch);
            break;
  case 'b': System.out.println(ch);
            break;
  case 'c': System.out.println(ch);
}
```

Execute this line

```
switch (ch) {
  case 'a': System.out.println(ch);
            break;
  case 'b': System.out.println(ch);
            break;
  case 'c': System.out.println(ch);
}
```

Execute next statement

```
switch (ch) {
  case 'a': System.out.println(ch);
            break;
  case 'b': System.out.println(ch);
            break;
  case 'c': System.out.println(ch);
}
```

Next statement;

Recall:

Write a program that reads a one-digit number from the user then prints on the screen the digit in letters.

```
int digit;
System.out.println("Enter a single digit number:");
digit = input.nextInt();


if (digit == 0){
            System.out.println(" ZERO ");
}else if (digit==1){
            System.out.println(" ONE ");
}else if (digit==2){
            System.out.println(" TWO ");
            …and so on….
}else{
            System.out.println(" Not a single-digit number");
      }
```

Slide provided by Thanaa Ghanem

Rewriting Example using `switch-case`

```
int digit;
System.out.println("Enter a single digit number");
digit = input.nextInt();
switch(digit)
{
        case 0:
                System.out.println("ZERO");
                break;
        case 1:
                System.out.println("ONE");
                break;
        case 2:
                System.out.println("TWO");
                break;
        …and so on …..
        default: System.out.println("Not a single digit number");
  }
```

Slide provided by Thanaa Ghanem

Switch program example

- Assume a clothing store might offer a discount that depends on the quality of the goods:
  - Class 'A' goods are not discounted at all.
  - Class 'B' goods are discounted 10%.
  - Class 'C' goods are discounted 20%.
  - anything else is discounted 30%.

```
double discount;
char code = 'B' ;
switch ( code ) {
        case 'A':
                discount = 0.0;
                break;
        case 'B':
                discount = 0.1;
                break;
        case 'C':
                discount = 0.2;
                break;
        default:
                discount =
0.3; }
```

Another switch program example

- Use a switch statement to write Java code to convert a number grade to a letter grade according to the following rules:
  - A: grade >= 90
  - B: grade >= 80
  - C: grade>=70
  - D: grade >=60
  - F: grade < 60

```
Scanner scan = new Scanner(System.in);
int grade;
System.out.println("Enter a number");
grade = scan.nextInt();


switch(grade/10){
  case 9:
  case 10:
      System.out.println("A");
      break;
  case 8:
      System.out.println("B");
      break;
   case 7:
        System.out.println("C");
        break;
   case 6:
       System.out.println("D");
       break;
    default:
       System.out.println("F");
       break;
  }
```

Slide provided by Thanaa Ghanem

Try it! Simple calculator implementation

- Create a Java project called Calculator application
- Create one class called Calculator that includes a main method.
- Inside the main method read three inputs from the user as follows:
  - Left operand
  - Operation (must be one of the following `+`,`-`,`*`,`/`, `or` `%`)
  - Right operand
- Use `switch` statement.
- The program then calculates the result of the operation and display the output. The program should display an error message if the user enters an unidentified operation.