CS1150
Principles of Computer Science
Boolean, Selection Statements (Part II)

Yanyan Zhuang

Department of Computer Science

http://www.cs.uccs.edu/~yzhuang

Review

- What's the scientific notation of 9,200,000?
- What's the value of a after each statement?
  - int a = 5;
  - a /= 3;
  - a--;

Review

- What's the scientific notation of 9,200,000?
  - 9.2e6
- What's the value of a after each statement?
  - int a = 5;
  - a /= 3; // a = 1
  - a--; // a = 0

Review

- Logical operators
  - !, &&, ||, ^
  - Mixing operators: p1 && p2 || p3, where p1, p2, p3 are boolean variables
  - x + y < 10 && x/y == 3 || z != 10 can be written as
    - ((x + y < 10) && (x/y == 3)) || (z != 10)
  - (!z) ||((x/y==0) && (x > 12))
  - How to test if x is between 10 and 20?
    - Can't write 10 <= x <= 20

Overview

- If statement
- Switch statement

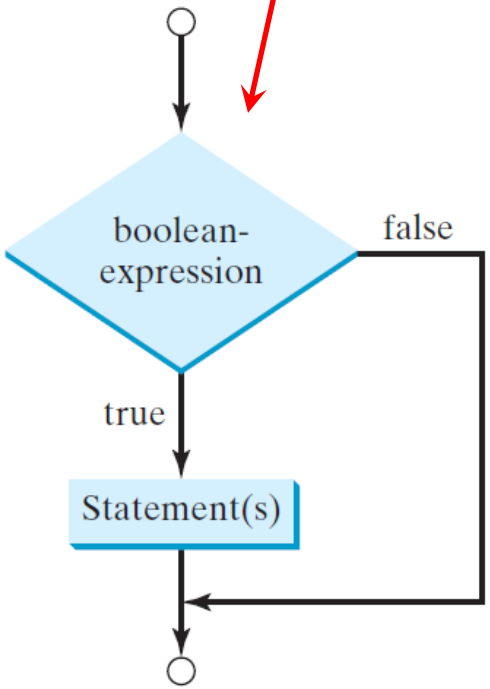
If Statement

- Select a path of execution based on a condition
  - A condition is evaluated (i.e. boolean expression)
  - If the condition is true
    - The statements in the true branch are executed
  - If the condition is false
    - The statements in false branch are executed (if there is an "else")

One-way if Statements

```
if (boolean-expression) {
  statement(s);
}
```

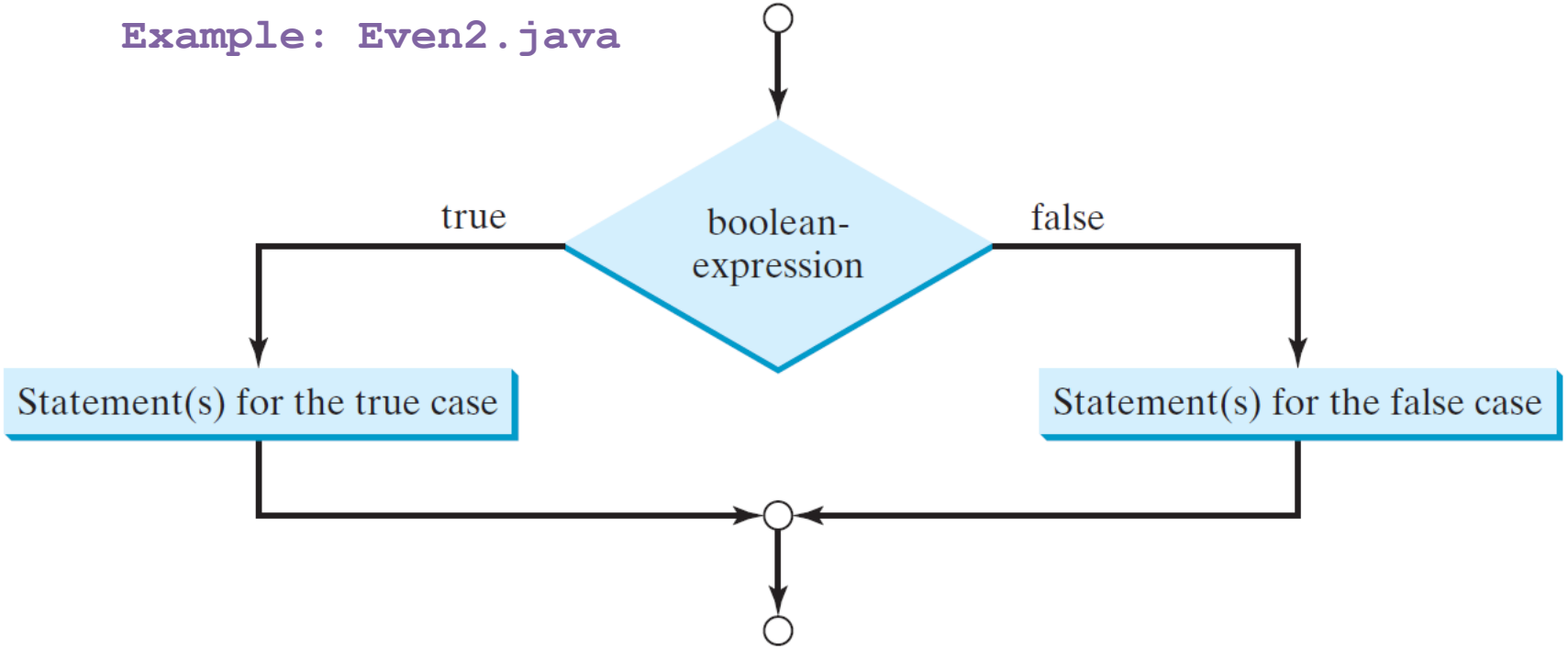
Example: Boolean2.java, Even1.java



Two-way if Statement

```
if (boolean-expression) {
  statement(s)-for-the-true-case;
}
else {
  statement(s)-for-the-false-case;
}
```

Example: Even2.java



What's wrong with this code?

```
Scanner input = new Scanner (System.in);
int mathGrade = input.nextInt();
String mathLetterGrade = "";   // "" is an empty String

if (mathGrade >= 90)
   mathLetterGrade = "A";
   System.out.println ("You got an A");
```

What's wrong with this code?

```
Scanner input = new Scanner (System.in);
int mathGrade = input.nextInt();
String mathLetterGrade = "";  // "" is an empty String


if (mathGrade >= 90); {
   mathLetterGrade = "A";
}
```

What's wrong with this code?

```
Scanner input = new Scanner (System.in);
int mathGrade = input.nextInt();


if (mathGrade = 90) {
  System.out.println ("You got 90 in your test!");
}
```

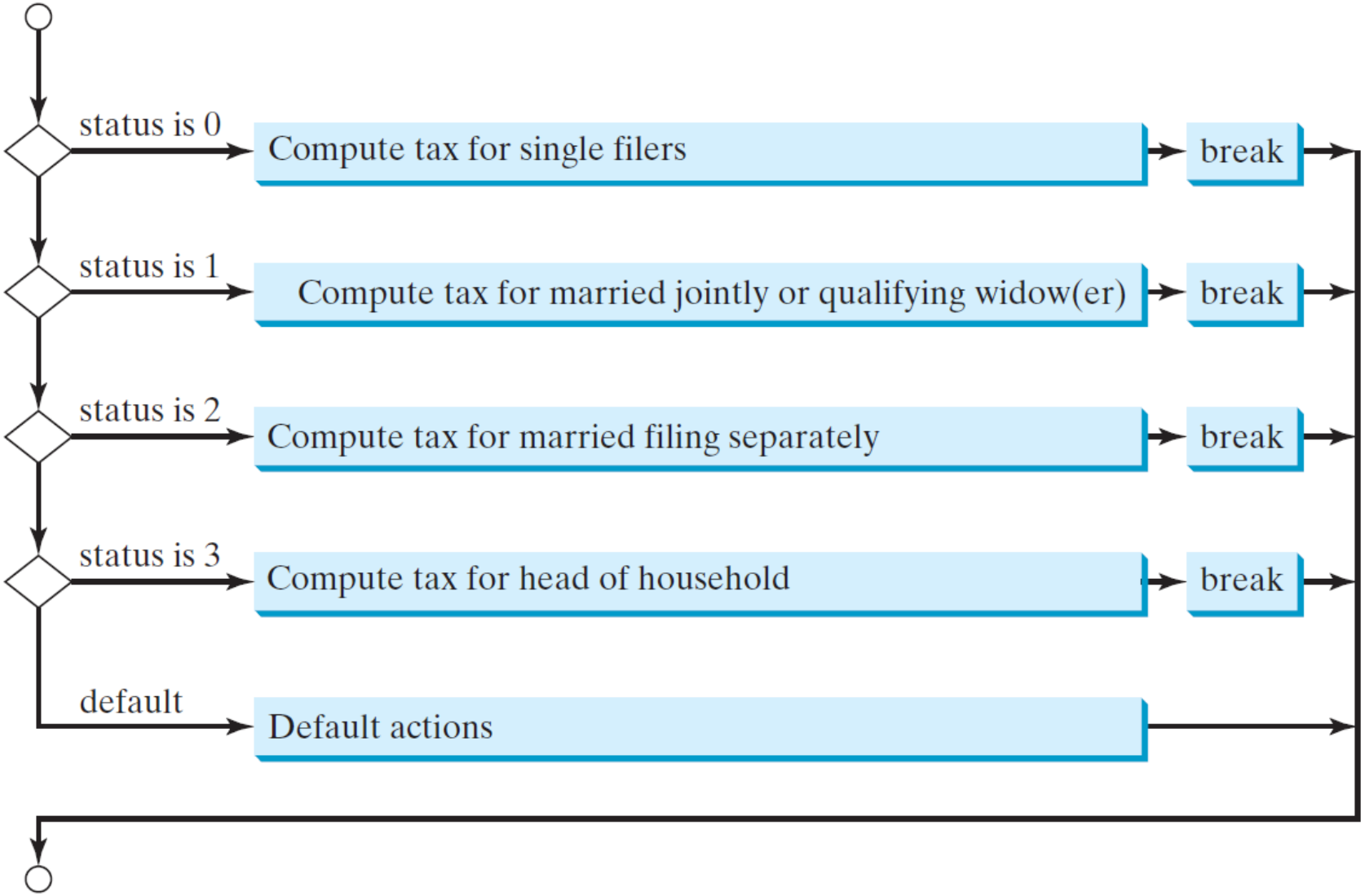
`switch` Statements

- Select a path of execution from a group of possibilities

Example: calculate tax returns for different status

```
switch (status) {
  case 0:  compute taxes for single filers;
           break;
  case 1:  compute taxes for married file jointly;
           break;
  case 2:  compute taxes for married file separately;
           break;
  case 3:  compute taxes for head of household;
           break;
  default: System.out.println("Errors: invalid status");
}
```

`switch` Statement Flow Chart



`switch` Statements

- How does it work?
  - The switch "expression" (e.g., status) is evaluated
  - If one of the cases "values" matches the value of the switch "expression"
    - The statements for that case are executed
    - Execution stops when a **break** statement is reached, or the end of the switch statement is reached
  - If no case "values" match the value of the switch "expression"
    - A default case is executed if it exists

```
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

The switch-expression must yield a value of char, byte, short, or int 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. The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression.

Note: value1, ..., and valueN are constant expressions, meaning that they **cannot** contain variables in the expression, such as 1 + x.

```
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

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 following case statements 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.

When the value in a **case** statement matches the value of the **switch-expression**, the statements *starting from this case* are executed until either a **break** statement or the end of the **switch** statement is reached.

`switch` Statement Notes

- switch expression
  - Must evaluate to a value of type char, byte, short, int
    - switch (x > 1) // Not allowed - evaluates to a boolean value
    - switch (x == 2) // Not allowed - another boolean expression

`switch` Statement Notes

- Case values
  - Are constants expressions
  - Cannot contain variables
    - case 0: System.out.println("......"); // valid
    - case (x+1): System.out.println("...."); // not valid
  - This is valid way to write cases

```
int value = 3;
switch (value) {
    case 1:case 2:case 3: System.out.println("case 1, 2, and 3"); break;
    case 4: System.out.println("case 4"); break;
    default: System.out.println("default");
}
```

`switch` Statement Notes

- break statement

```
int day = 2;
switch (day) {
        case 1:
        case 2:
        case 3:
        case 4:
        case 5: System.out.println("Weekday"); break;
        case 0:
        case 6: System.out.println("Weekend");
}
```

Conditional Expressions

- Shortcut way to write a two-way if statement (if-else)
  - Consists of the symbols ? and : (aka the "ternary" operator)
  - result = expression ? value1 : value2;
    - expression can be either a boolean value or a statement that evaluates to a boolean value
    - The conditional "expression" is evaluated
    - If the expression is true, value1 is returned
    - If the expression is false, value2 is returned

Conditional Expressions

- if (a < 0) {

  absValue = -a;

  }

  else {

  absValue = a;

  }

- Can be re-written as:

  absValue = (a < 0) ? -a : a;

Let's do an exercise!

- Ordering lunch
  - Please choose side: salad ($4) or fries ($3.5)
    - If invalid choice, exit program
  - Please choose entrée: chicken ($9), beef ($12) or fish ($11)
    - If invalid choice, exit program
  - Calculate subtotal with CO tax (8.25%)
  - Tipping? 0%, 10%, 15% or 20%
    - If invalid choice, exit program
  - Print total amount due (subtotal + tax + tipping)
    - Two decimal points

More on ++ and --

- Placement of prefix or postfix cause different results when **in expressions** so be careful!!!

```
int x = 1;
int y = 3;
y = ++x;   // do increment then assignment => y is 2, x is 2

int x = 1;
int y = 3;
y = x++;  // do assignment then increment => y is 1, x is 2
```

More on ++ and --

- Simple (but confusing) example
  - int numDays = 100;
  - System.out.println (numDays);
  - System.out.println (++numDays); // numDays=101, print 101
  - System.out.println (numDays++); // print 101, numDays=102
  - System.out.println (numDays);
- Recommendation: avoid ++ and -- in expressions (to avoid confusion, make code more readable); used in isolation is not a problem

Summary

- Boolean data type
- If statement
- Switch statement