CISC 3120
C05: Flow Controls

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Recap and issues
  - Selections & iterations
- Flow controls in Java
  - Booleans and Conditions
  - More on selection & iterations
  - Break, continue, and return
- Assignments

Recap & Observations

- Programming is also a skill
- What is your learning style?
- Concept of a software project
- What should be in a repository?

Programming is a skill too

- Programming is learned by programming, not from reading books.
  - Translate specification to algorithm, translate algorithm to code

Learning Style

- Your most comfortable way of learning may not be your best
  - Work individually?
  - Work in a group?
  - Ask questions?
  - Passively or proactively?

Software Project

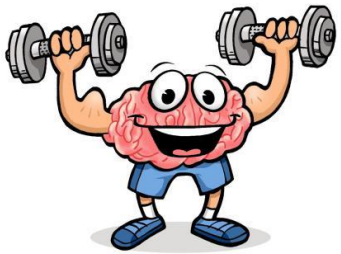
- A software project consists of many pieces
  - Example
    - A Java project consists of many Java classes.
- An IDE helps you manage software projects

SCM Repository

- What should be in the code repository?
  - Derived artifacts are usually NOT in the repository.
    - Example:
      - For Maven project,
        - .settings, .classpath, .project
      - For any Java project
        - .classes, target, bin

Questions?

- Questions?
- Suggestions?

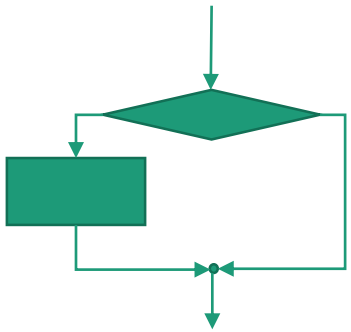


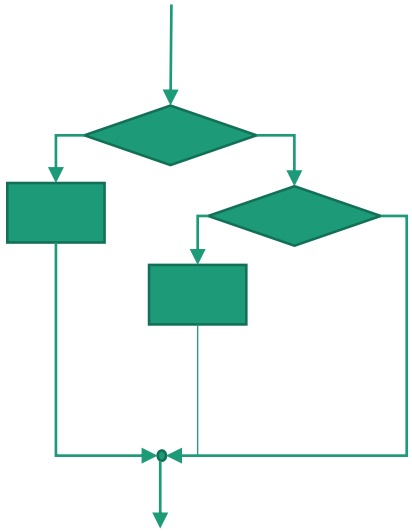
If you understand what you're doing, you're not learning anything.

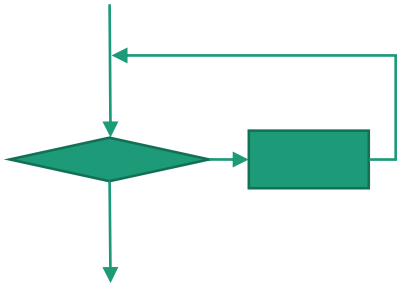
-- Anonymous

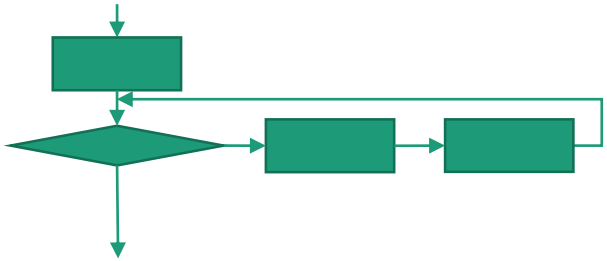
Recap: Selections & Iterations

- Selections
  - if-then statement
  - if-then-else statement
- Iterations
  - while statement
  - for statement



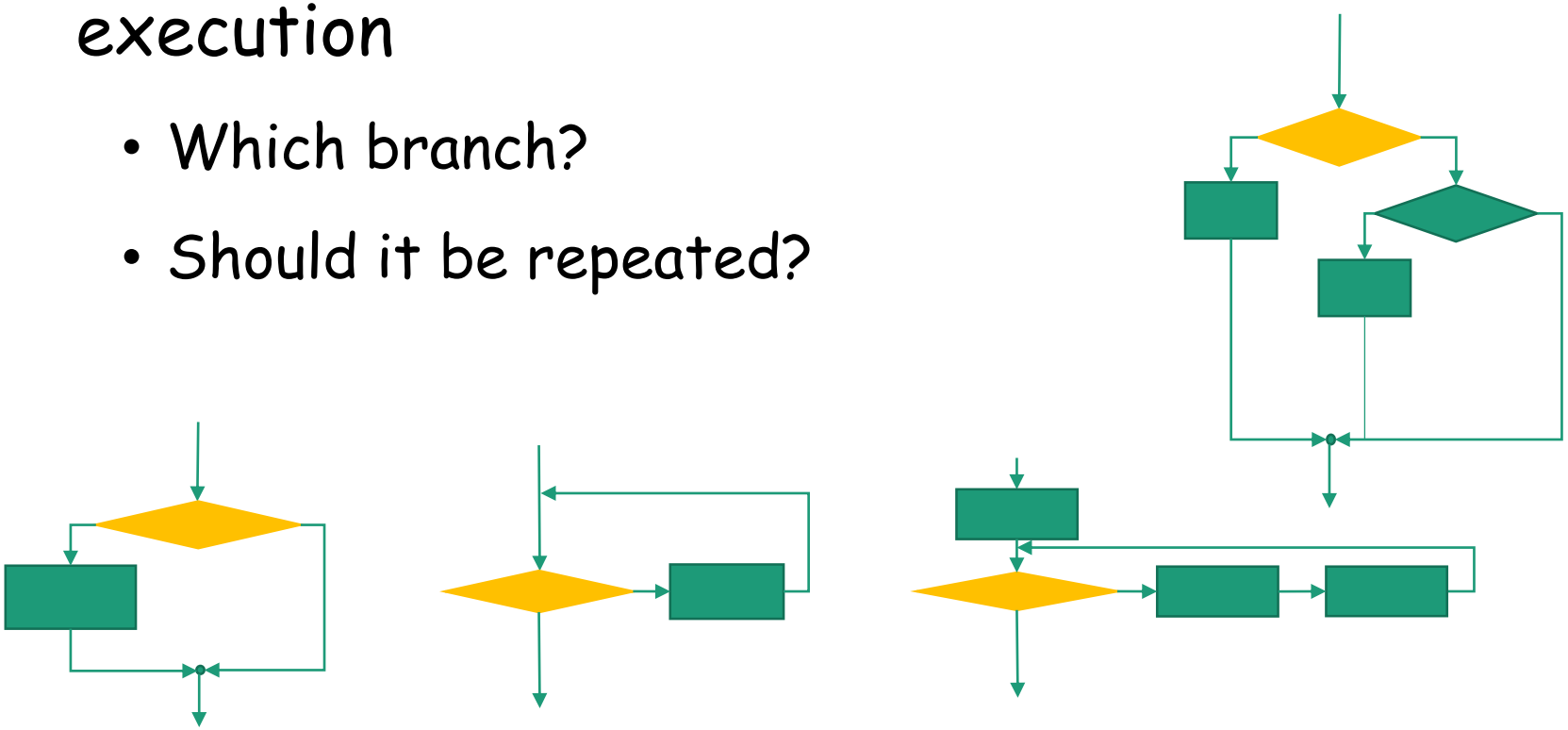






Boolean Expression

- Often used to control the flow of program execution
  - Which branch?
  - Should it be repeated?



Boolean Expression

- An expression evaluates to either true or false
  - Primitive data type: boolean
  - true and false are Java keywords
  - Relational operators and conditional operators
  - Observe a few examples in FlowControlExamples in the SamplePrograms repository
    - BooleanConditionsExamples.java

Relational Operators

- <, <=, >, >=, ==, !=
- Where you can use them depending on data types
  - Is it meaningful to say "less than" or "greater than") for the data type?
  - In Java
    - <, <=, >, >= are used with numerical values and variables
    - == and != can be used with both primitive data types and reference types (objects)

Testing Object Equivalence

- Relational operators "==" and "!=" can be used to compare objects

```
Integer n1 = new Integer(3120);
Integer n2 = new Integer(3120);
System.out.println(n1 == n2);
System.out.println(n1 != n2);
System.out.println(n1.equals(n2));
```

  - Observe it in FlowControlExamples in the SamplePrograms repository

Two Dogs are Equal?

- How about objects of your classes?
  - Test two dogs being equal?

```
class Dog {
   boolean equals(Dog other) {
      ……
   }
}
```

  - See the TwoEqualDogs class

Advanced: Two Dogs are Equal?

- Proper way to do it (to be discussed in the future)
  - Test two dogs being equal?

```
class Dog {
  @Override
  public boolean equals(Object other) {
     ……
  }
  @Override
  public int hashCode() {
    ……
  }
}
```

How about String Objects?

- Strings are objects. How about these? Any surprises?

```
String s1 = new String("CISC 3120");
String s2 = new String("CISC 3120");
System.out.println(s1 == s2);
System.out.println(s1.equals(s2));

String s1 = "CISC 3120";
String s2 = "CISC 3120";
System.out.println(s1 == s2);
System.out.println(s1.equals(s2));
```

More Examples on Strings

- Compare these two:

```
String s1 = "CISC 3120";
String s2 = "CISC 3120";
System.out.println(s1 == s2);
System.out.println(s1.equals(s2));

 // string literals allocated in run-time constant pool
String s1 = "CISC 3120";
String s2 = s1;
System.out.println(s1 == s2);
System.out.println(s1.equals(s2));
```

Conditional Operators

- Three operators that produce a Boolean value
  - AND (&&)
  - OR (||)
  - NOT (!)
- Do NOT confuse them with
  - Bitwise AND (&)
  - Bitwise OR (|)
  - Bitwise NOT (~)
  - Bitwise XOR (^)

Short-Circuiting

- JVM ceases to evaluate further once a truth or a falsehood value is unambiguously determined.
- Example in FlowControlExamples in the SampleProgram repository

Questions

- Boolean data type and Boolean values
- Boolean expressions
- Conditional operations
- Short-circuiting

Selections (Branching)

- If-then
- If-then-else
- Switch
- Examples
  - SelectionExamples.java

Switch Statement

- Form

```
switch(integral-selector) {
    case integral-value1 : statement; break;
    case integral-value2 : statement; break;
    case integral-value3 : statement; break;
    case integral-value4 : statement; break;
    case integral-value5 : statement; break;
    // ...
    default: statement;
}
```

Questions?

- Selections in flow control

Iterations

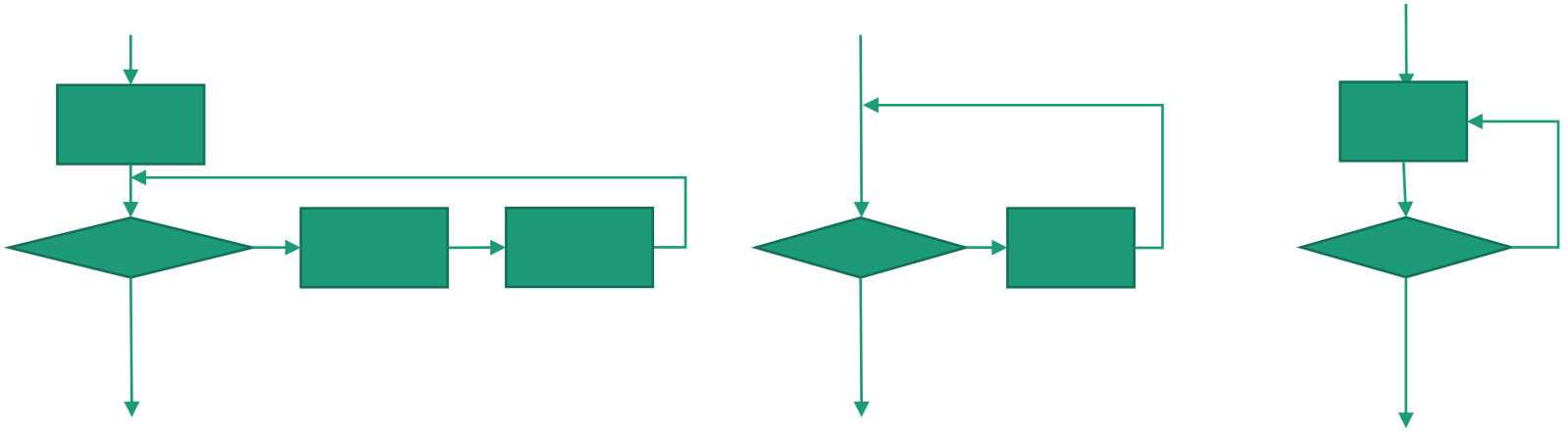
- while statement
- for statement
- enhanced for statement
- do-while statement
- Examples
  - IterationExamples.java

while, for, and do-while

- An algorithm can be implemented using either with care
- However, one may be more conveniently to use than the other

while, for, and do-while

- Which flow chart corresponds to while, for, and do-while?



Questions?

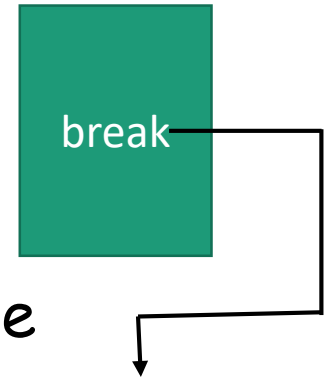
- Iterations in flow control

Break, Continue, and Return

- Break
- Continue
- Return
- Examples
  - IterationExamples.java
  - SelectionExamples.java

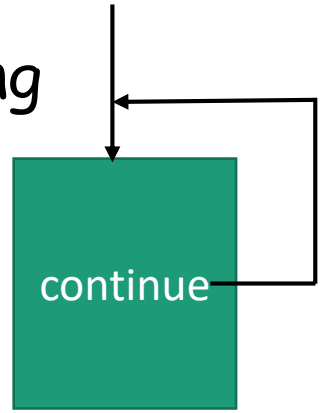
Break

- A break statement
  - transfer controls to the innermost enclosing break target
  - then immediately completes it normally.
- Typical break target
  - switch, while, do, or for statement of the immediately enclosing method



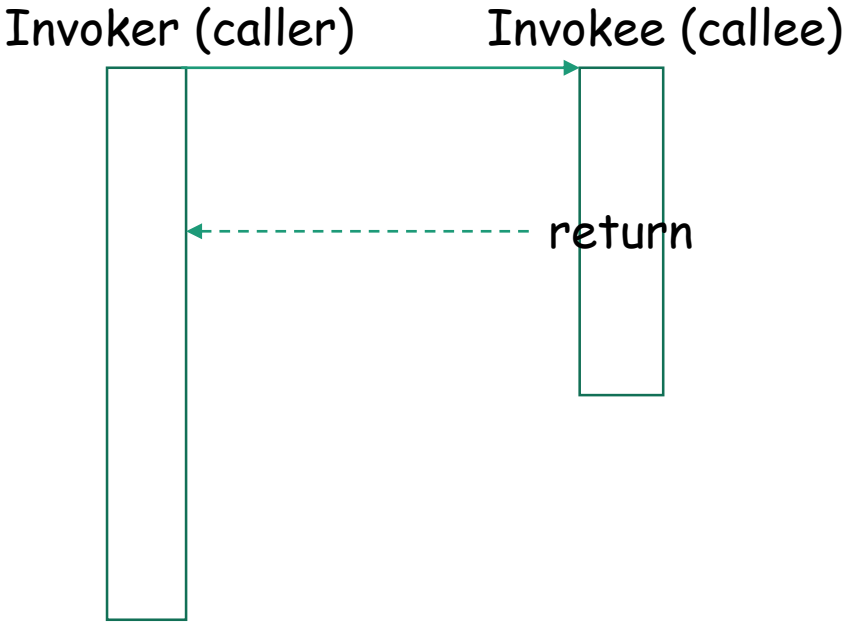
Continue

- A continue statement
  - transfers control to the innermost enclosing continue target
  - immediately ends the current iteration and begins a new one.
- Typical continue target
  - while, do, or for statement of the immediately enclosing method, constructor



Return

- A return statement returns control to the invoker of a method.



Question

- More discussions on
  - Selection & iterations
- Discussion on the break, continue, and return statements.

Console Input/Output

- Standard output
  - System.out
    - print, println
- Standard input
  - System.in
- Standard error
  - Report errors
  - System.err
    - print, println

Console Output

- Standard output and standard error
- Often use String operations, such as, String concatenation (the "+" operator)
- Examples

```
int n = (int) Math.random() * 10;
double sqrt = Math.sqrt(n);
String m = "Output: ";
System.out.println(n);                  // value of variable n
System.out.println("n = " + n);         // with a label "n = "
System.out.println(m + "n = " + n); // with a String variable
System.out.println("The square root of " + n + " is " + sqrt);
```

Console Input

- System.in is low-level (read one character at a time). Wrap it with classes for input.
- Use Scanner

```
Scanner scanner = new Scanner(System.in);

double d = scanner.nextDouble();
```

- Examples:
  - ConsoleIOExamples.java

Questions

- Console input/output
- Simple use cases of the Scanner class

Assignments

- Practice assignment
- CodeLab assignment
- Upcoming: project 1