Lecture 4
Control Structures

MIT – AITI 2004
What are Control Structures?

• Java will execute your code in a specific sequence
• Control structures are a way to alter the natural sequence of execution in a Java program
• In other words, they act as “direction signals” to control the path a program takes
• Control structures include:
  • block statements
  • decision statements
  • loops
Decision Statements
If Statements

• The “if” decision statement executes a statement conditionally

```java
if (expression) {
    statement;
} else {
    next_statement;
}
```

• The expression must produce a boolean value, either true or false

• If expression returns true, statement is executed and then next_statement

• If expression returns false, statement is not executed and the program continues at next_statement
if (expression)
  statement
next_statement
If-Else Statements

• The basic “if” statement can be extended by adding the “else” clause

```java
if (expression) {
    statement1;
} else{
    statement2;
} next_statement;
```

• Again, the expression must produce a boolean value

• If expression returns true, statement1 is executed and then next_statement is executed.

• If expression returns false, statement2 is executed and then next_statement is executed.
if (expression) {
  statement1
} else {
  statement2
}
next_statement
Here is an example of chained if-else statements:

```java
if (grade == 'A')
    System.out.println("You got an A.");
else if (grade == 'B')
    System.out.println("You got a B.");
else if (grade == 'C')
    System.out.println("You got a C.");
else
    System.out.println("You got an F.");
```
Switch Statements

- The `switch` statement enables you to test several cases generated by a given expression.
- The expression must produce a result of type `char`, `byte`, `short` or `int`, **but not** `long`, `float`, or `double`.
- For example:
  ```java
  switch (expression) {
      case value1:
          statement1;
      case value2:
          statement2;
      default:
          default_statement;
  }
  ```
  Every statement after the true case is executed
switch (expression) {
  case value1:
    // Do value1 thing
    break;
  case value2:
    // Do value2 thing
    break;
  ...
  default:
    // Do default action
    break;
}
// Continue the program
Break Statements in Switch Statements

• The break; statement tells the computer to exit the switch statement
• For example:

```java
switch (expression) {
    case value1:
        statement1;
        break;

    case value2:
        statement2;
        break;

    default:
        default_statement;
        break;
}
```
switch (expression){
    case value1:
        // Do value1 thing
        break;
    case value2:
        // Do value2 thing
        break;
    ... 
    default:
        // Do default action
        break;
}
// Continue the program

// Continue the program
Remember the Example…

• Here is an example of chained if-else statements:

```java
if (grade == 'A')
    System.out.println("You got an A.");
else if (grade == 'B')
    System.out.println("You got a B.");
else if (grade == 'C')
    System.out.println("You got a C.");
else
    System.out.println("You got an F.");
```
• Another way to do this same example is to use the switch statement
• Complicated if-else chains can be rewritten with the switch statement

    switch (grade) {
        case 'A':
            System.out.println("You got an A.");
            break;
        case 'B':
            System.out.println("You got a B.");
            break;
        case 'C':
            System.out.println("You got a C.");
            break;
        default:
            System.out.println("You got an F.");
    }
Loops
• A loop allows you to execute a statement or block of statements repeatedly.

• There are three types of loops in Java:
  1. for loops
  2. while loops
  3. Do-while loops (will not discuss in this course)
The for Loop

```
for (initialization_expression;
    loop_condition;
    increment_expression){
    //statement
}
```

- The control of the for loop appear in parentheses and is made up of three parts.

1. The first part, the `initialization_expression`, sets the initial conditions for the loop and is executed before the loop starts.

2. Loop executes so long as the `loop_condition` is true and exits otherwise.

3. The third part of the control information, the `increment_expression`, is usually used to increment the loop counter. This is executed at the end of each loop iteration.
• For example:
  ```
  int limit = 5;
  int sum = 0;
  for(int i = 1; i<=limit; i++){

    /* initialization_expression
      loop_condition
      increment_expression */

    sum += 2;

  }
  ```

• What is the value of `sum`?
  10
• Another example:

```java
for(int div = 0; div<1000; div++){
    if(div % 12 == 0){
        System.out.println(div+"is divisible by 12");
    }
}
```

• This loop will display every number from 0 to 999 that is evenly divisible by 12.
• If there is more than one variable to set up or increment they are separated by a comma.

```java
for (i=0, j=0; i*j<1000; i++, j+=2) {
    System.out.println(i+"*"+j+"="+i*j);
}
```

• You do not have to fill every part of the control of the `for` loop but you must still have two semi-colons.

```java
for (int i=0; i<100; ) {
    sum+=i;
    i+=i;
}
```
The **while** Loop

```java
while (expression)
{
    statement
}
```

- This while loop executes as long as the given logical expression between parentheses is **true**. When expression is **false**, execution continues with the statement following the loop block.
- The expression is tested at the beginning of the loop, so if it is initially **false**, the loop will not be executed at all.
• For example:

```java
int limit = 7;
int sum = 0;
int i = 1;

while (i < limit){
    sum += i;
    i++;
}
```

• What is the value of `sum`?

21
The **for** loop

1. Initialize count
2. Test condition is true?
   - **n**
   - **y**
   - Execute loop statement(s)
   - Increment count
3. New statement

The **while** loop

1. Test condition is true?
   - **n**
   - **y**
   - Execute loop statement(?)
2. Next statement
The continue Statement

• The continue statement causes the loop to exit its current “trip” through the loop and start over at the first statement of the loop. Here are two examples:

Example 1:

```java
int index = 0;
while (index <= 5) {
    index += 1;
    System.out.println("The index is "+index);
    if (index == 4) {
        continue;
    }
}
```

The index is 1
The index is 2
The index is 3
The index is 5
The index is 6
The index is 6
Example 2:

```java
int sum = 0;
for(int i=1; i<=6; i++){
    if(i%3==0){
        continue;
    }
    sum += i;
}
```

What is the value of sum?

12
Using the break Statement in Loops

- We have seen the use of the break statement in the switch statement.
- In loops, you can use the break statement to exit the current loop you are in. Here is an example:

```java
int index = 0;
while (index <= 10) {
    index++;
    if (index == 3)
        break;
    System.out.println("The index is" + index);
}
```
Nested Loops

• You can nest loops of any kind one inside another to any depth. Here is an example:

```java
int totalCount = 0;
while (totalCount < 20) {
    for (int i = 0; i < 2; i++) {
        totalCount += i;
    }
}
System.out.println(totalCount);
```

20
• Final example:

```java
for (int i = 1; i <= 10; i++) {
    if (i % 2 == 0) {
        System.out.println("The number " + i + " is even.");
    } else {
        System.out.println("The number " + i + " is odd.");
    }
}
```
1. In the switch statement expression must produce a result of what type?  
   char, byte, short, int

2. What must be used to separate each section of a for statement.

3. Which statement causes a program to go back to the statement that began a loop and then keep going from there.

4. Write a for loop that outputs 100→1 in reverse sequence.

5. Write a for loop that outputs all numbers that are divisible by 3 between 0-50.

Answers:
1. From slide “Switch Statements”: char, byte, short or int
2. From slide “The for loop”: a semi-colon
3. From slide “The continue Statement”: continue
4. for(int i=100; i>0; i--)
   System.out.println(i);
5. for(int i=0; i<=50; i++) {
   if (i%3 == 0)
      System.out.println(i);
}
1. In the `switch` statement expression must produce a result of what type?
   - `char, byte, short, int`

2. What must be used to separate each section of a `for` statement?
   - `; (semi-colon)`

3. Which statement causes a program to go back to the statement that began a loop and then keep going from there?

4. Write a `for` loop that outputs 100→1 in reverse sequence.

5. Write a `for` loop that outputs all numbers that are divisible by 3 between 0-50.
POP QUIZ

• In the switch statement expression must produce a result of what type?  
  char, byte, short, int

• What must be used to separate each section of a for statement.  
  ; (semi-colon)

• Which statement causes a program to go back to the statement that began a loop and then keep going from there.  
  continue

• Write a for loop that outputs 100→1 in reverse sequence.

• Write a for loop that outputs all numbers that are divisible by 3 between 0-50.
**POP QUIZ**

- In the switch statement expression must produce a result of what type? `char, byte, short, int`

- What must be used to separate each section of a `for` statement? `; (semi-colon)`

- Which statement causes a program to go back to the statement that began a loop and then keep going from there? `continue`

- Write a `for` loop that outputs `100 → 1` in reverse sequence.

- Write a `for` loop that outputs all numbers that are divisible by 3 between 0-50.

```java
for(int i=100; i>=0;i--) {
    System.out.println(i);
}
```
POPC QUIZ

• In the switch statement expression must produce a result of what type?  
  char, byte, short, int

• What must be used to separate each section of a for statement.  
  ; (semi-colon)

• Which statement causes a program to go back to the statement that began a loop and then keep going from there.  
  continue

• Write a for loop that outputs 100→1 in reverse sequence.
  for(int i=100; i>=0;i--) {  
    System.out.println(i);}

• Write a for loop that outputs all numbers that are divisible by 3 between 0-50.
  for(int i=0;i<=50;i++) {
    if(i%3 == 0) { SOP(i); }
  }