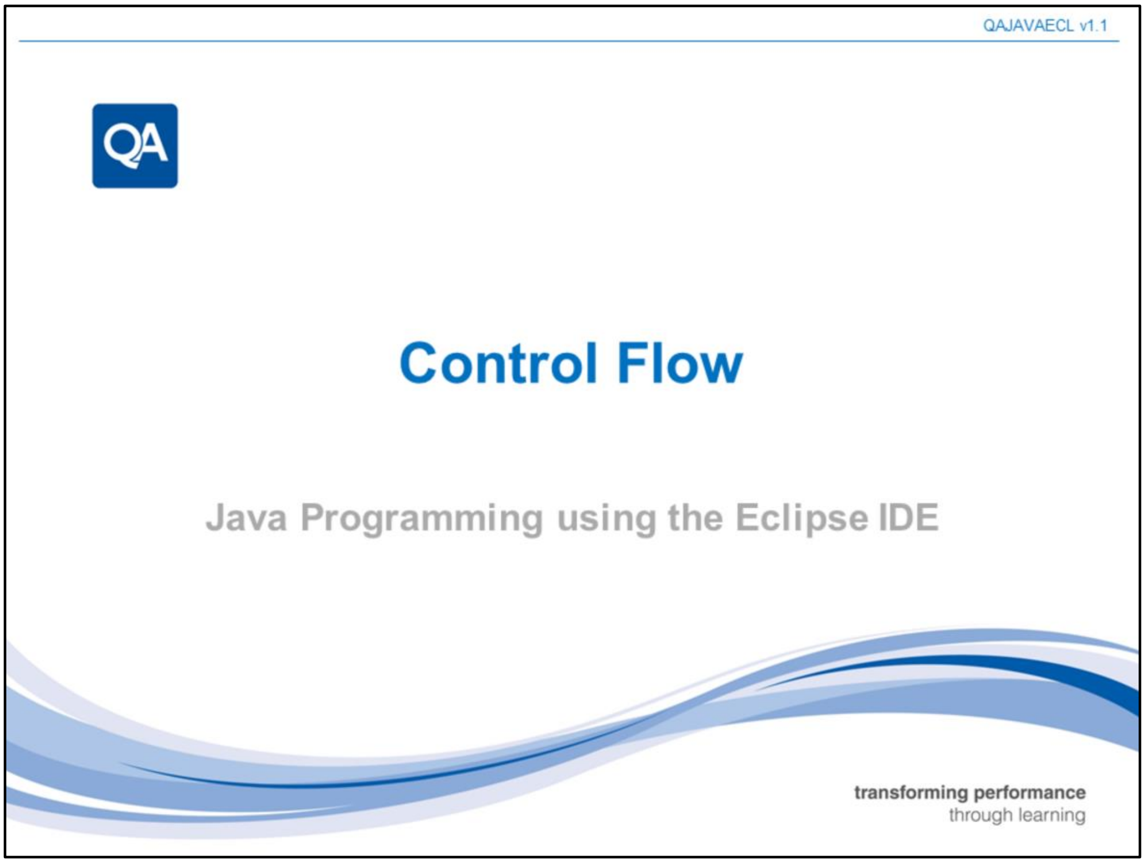
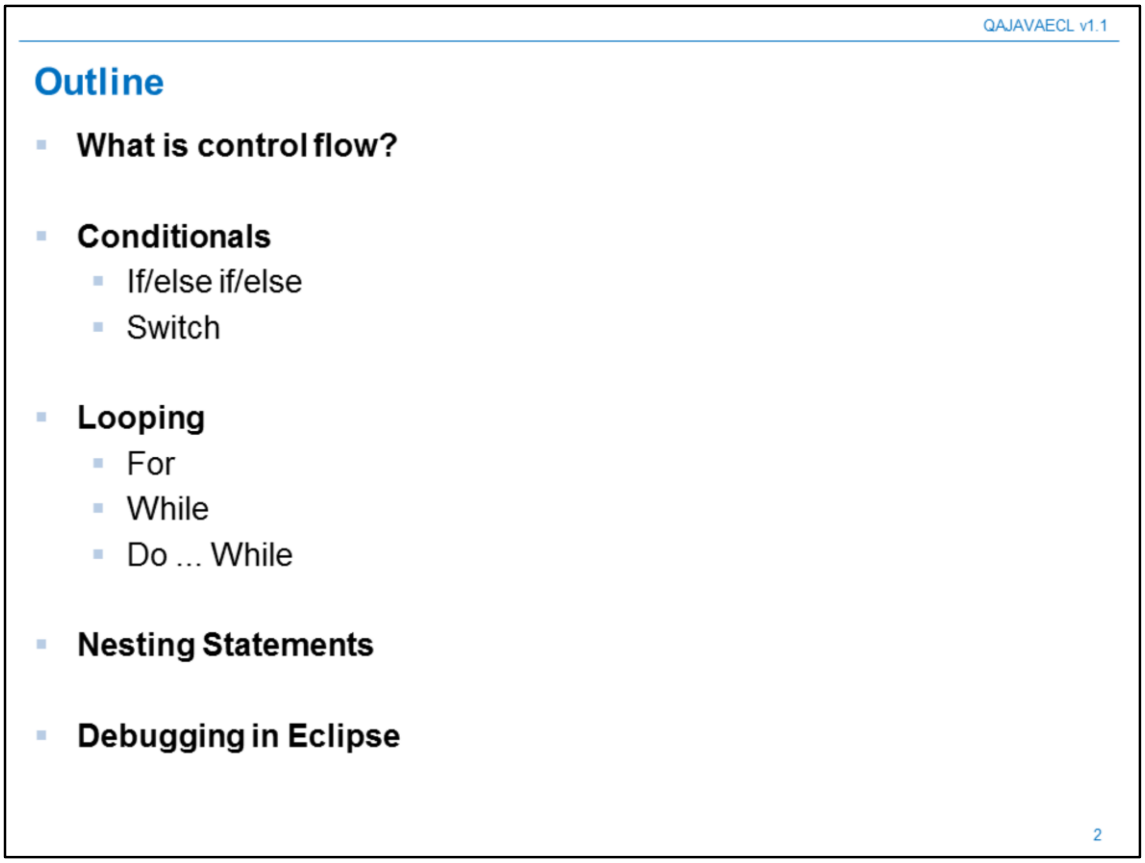


QAJAVAECL v1.1

Control Flow

Java Programming using the Eclipse IDE

transforming performance
through learning

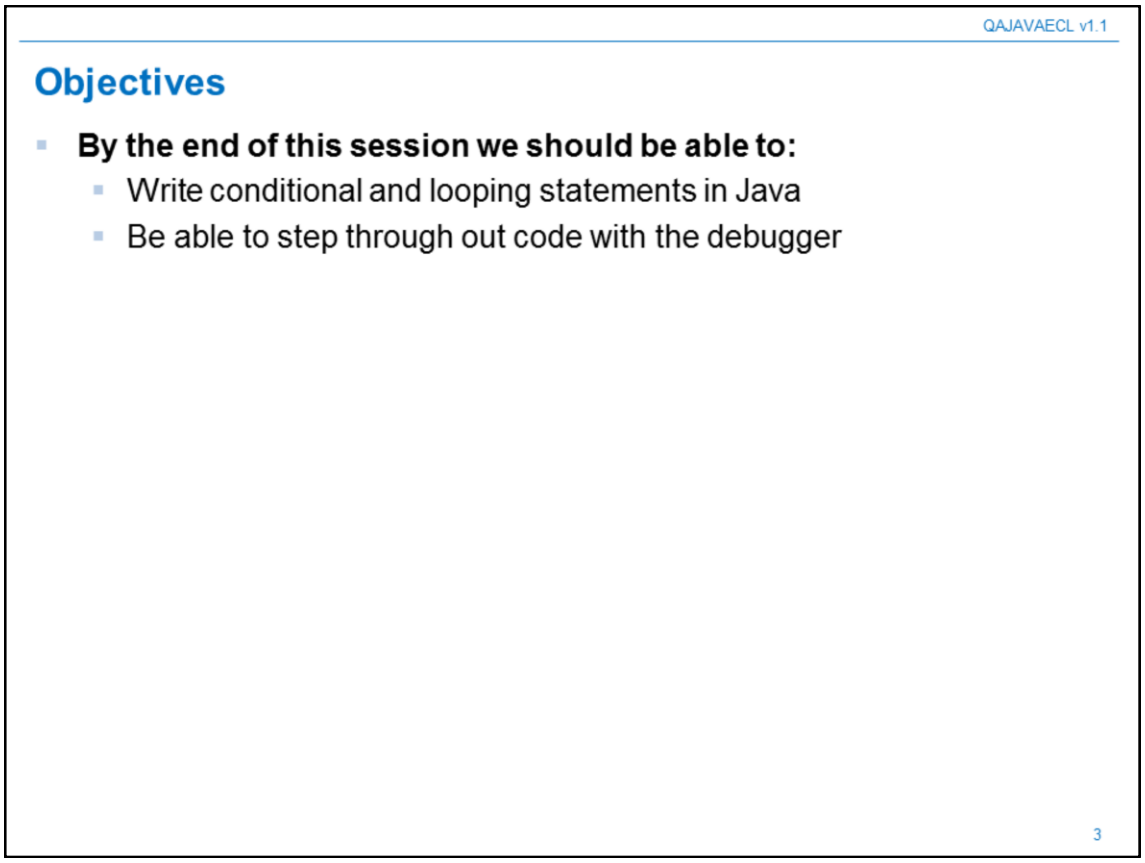


QAJAVAECL v1.1

Outline

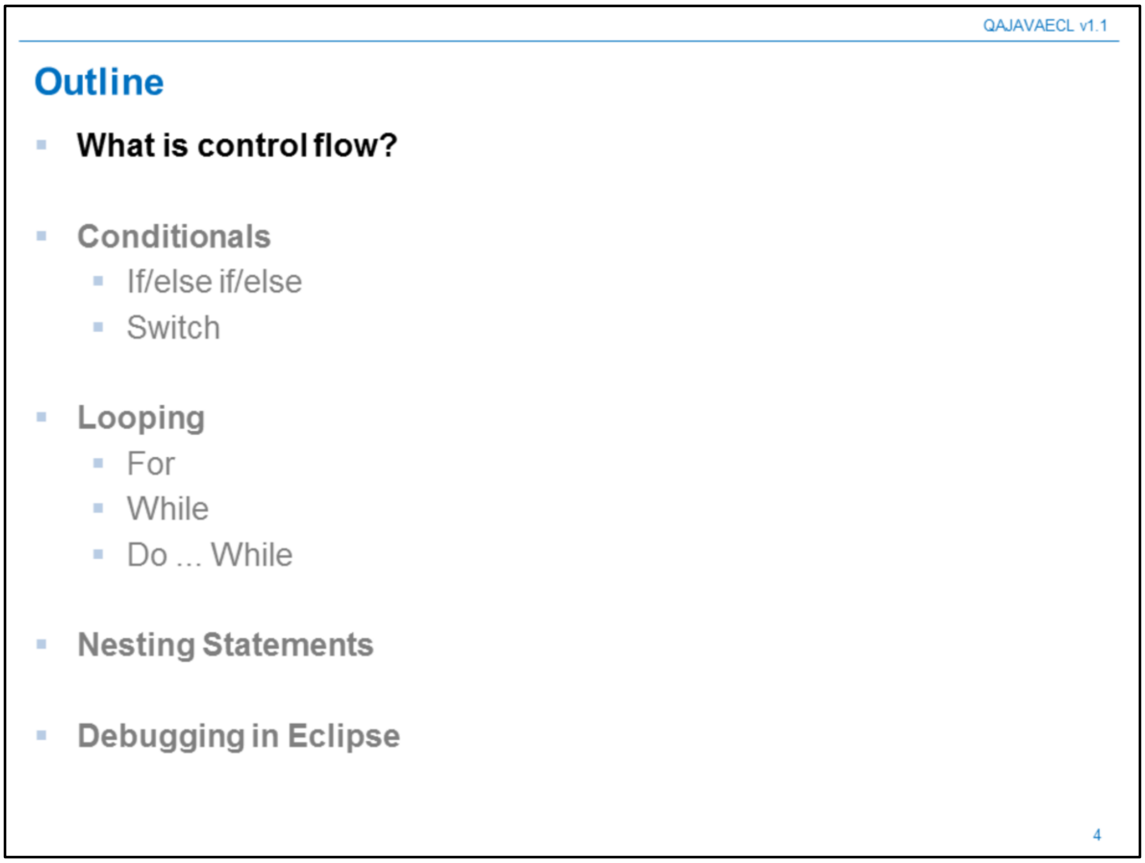
- **What is control flow?**
- **Conditionals**
  - If/else if/else
  - Switch
- **Looping**
  - For
  - While
  - Do ... While
- **Nesting Statements**
- **Debugging in Eclipse**

2



Objectives

- **By the end of this session we should be able to:**
  - Write conditional and looping statements in Java
  - Be able to step through out code with the debugger

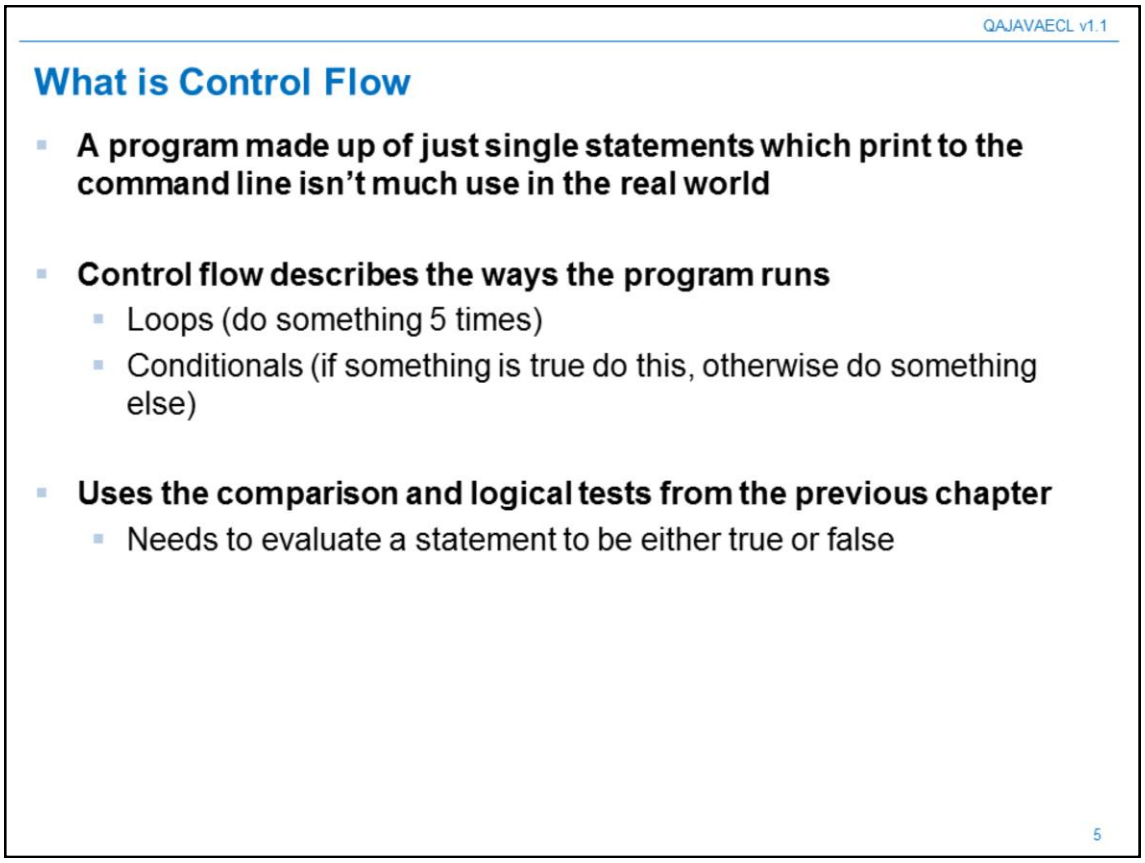


QAJAVAECL v1.1

Outline

- **What is control flow?**
- **Conditionals**
  - If/else if/else
  - Switch
- **Looping**
  - For
  - While
  - Do ... While
- **Nesting Statements**
- **Debugging in Eclipse**

4



QAJAVAECL v1.1

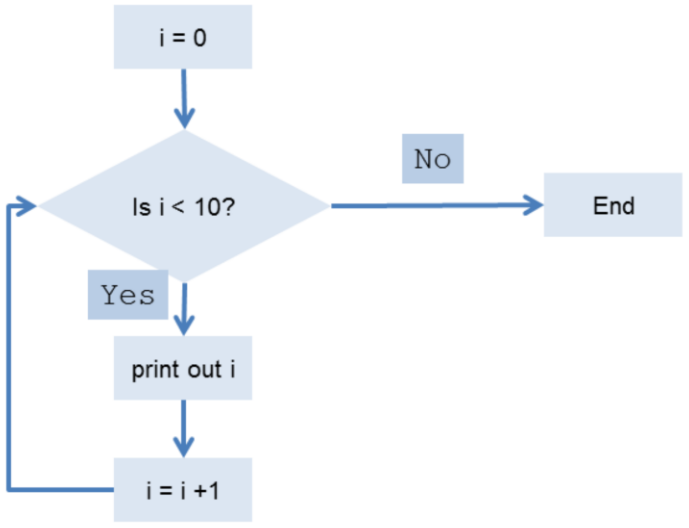
What is Control Flow

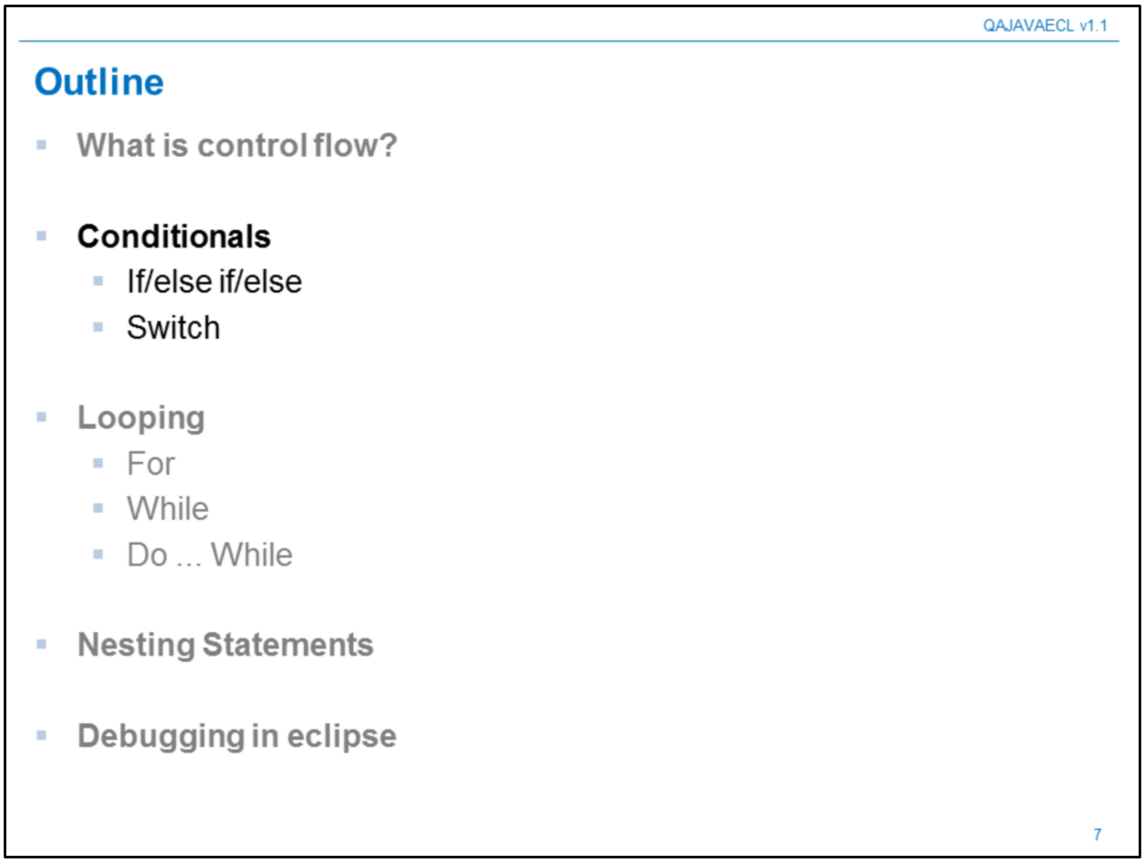
- **A program made up of just single statements which print to the command line isn't much use in the real world**
- **Control flow describes the ways the program runs**
  - Loops (do something 5 times)
  - Conditionals (if something is true do this, otherwise do something else)
- **Uses the comparison and logical tests from the previous chapter**
  - Needs to evaluate a statement to be either true or false

5

Flow Diagrams

- **Programs are a series of instructions**
  - We can draw these as flow diagrams to try and understand what is going on



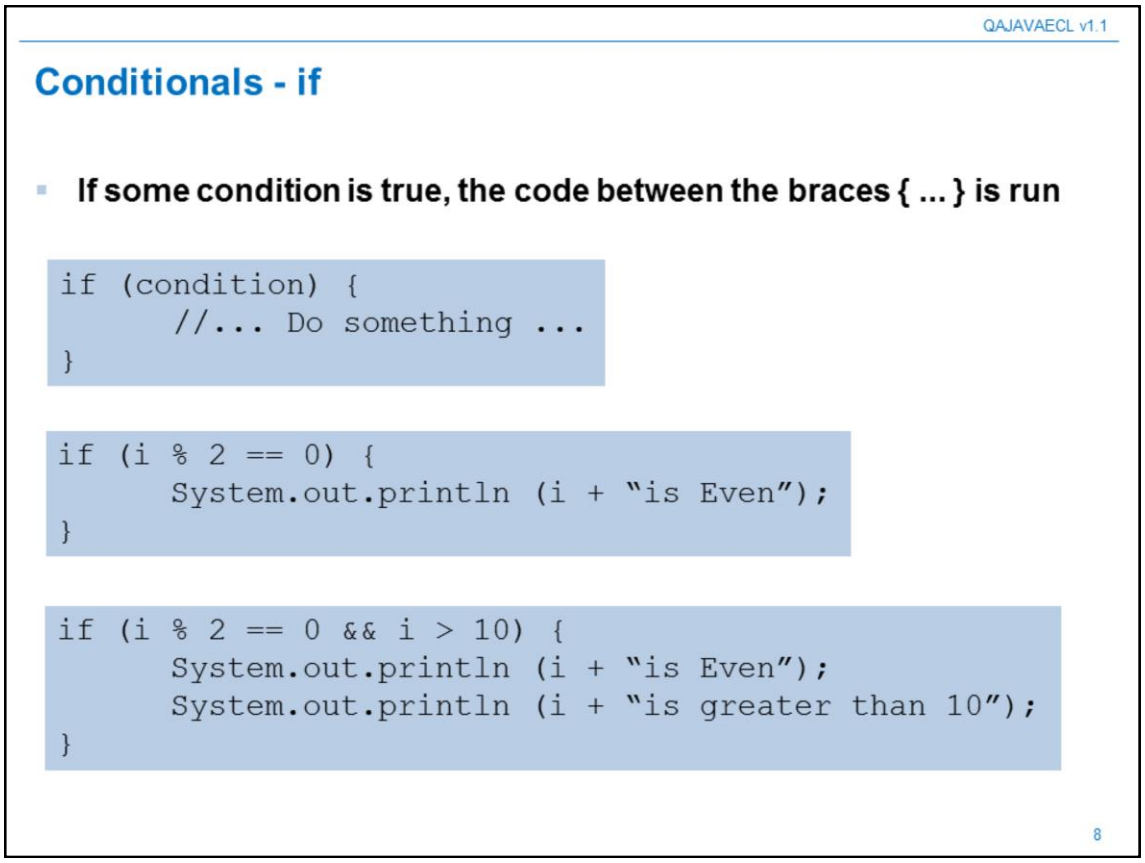


QAJAVAECL v1.1

Outline

- What is control flow?
- **Conditionals**
  - If/else if/else
  - Switch
- Looping
  - For
  - While
  - Do ... While
- Nesting Statements
- Debugging in eclipse

7



QAJAVAECL v1.1

Conditionals - if

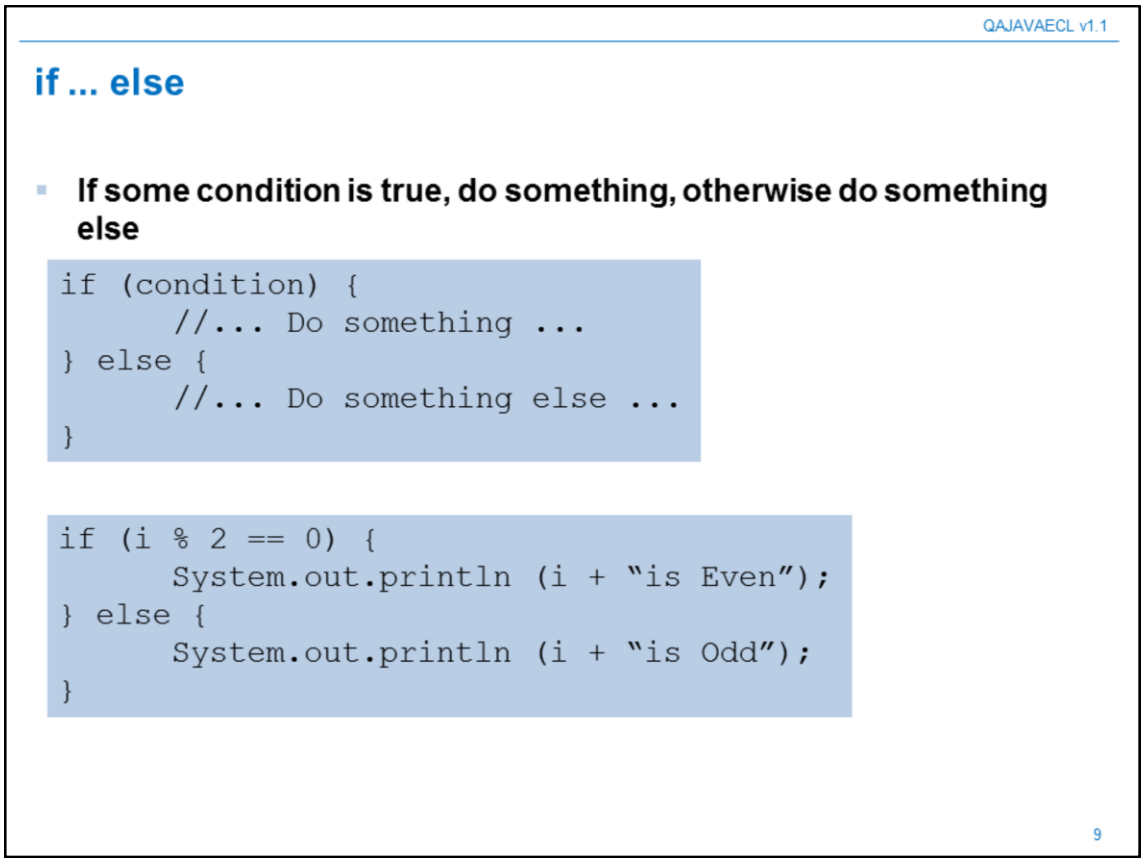
- If some condition is true, the code between the braces { ... } is run

```
if (condition) {
      //... Do something ...
}
```

```
if (i % 2 == 0) {
      System.out.println (i + "is Even");
}
```

```
if (i % 2 == 0 && i > 10) {
      System.out.println (i + "is Even");
      System.out.println (i + "is greater than 10");
}
```

8

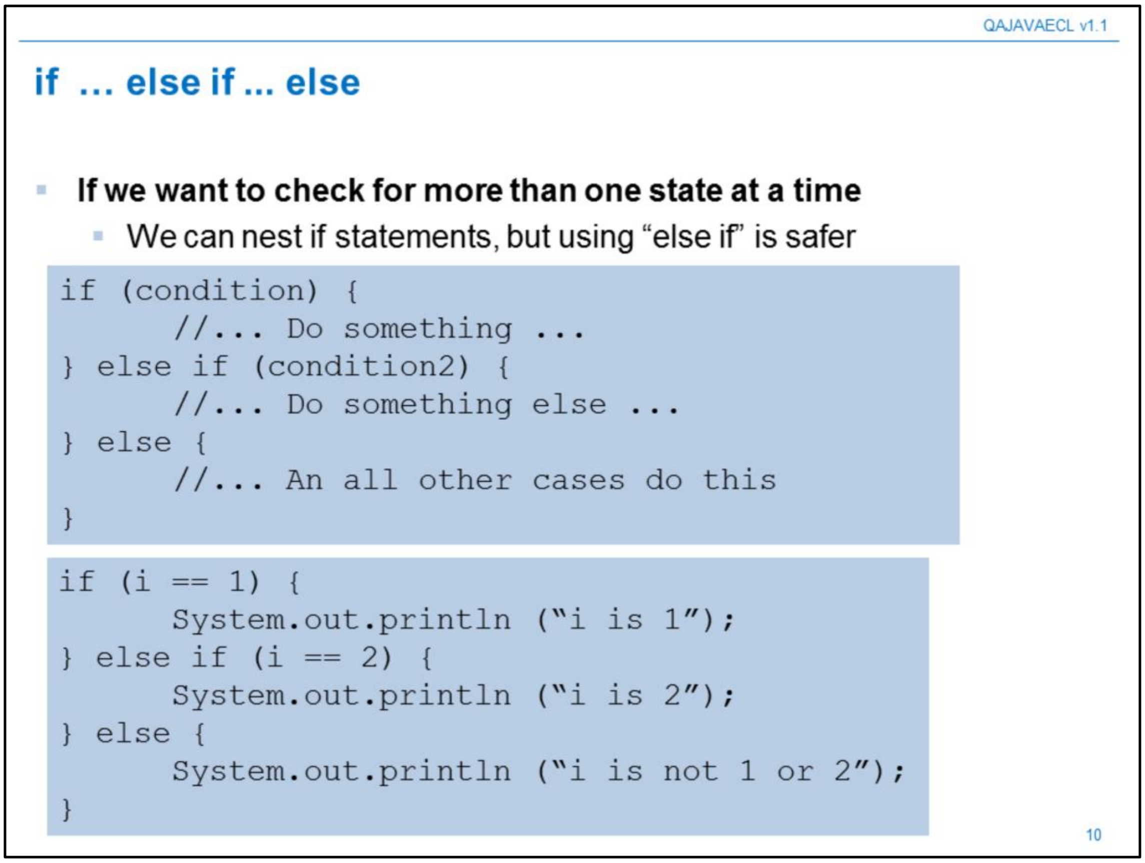


if … else

- If some condition is true, do something, otherwise do something else

```
if (condition) {
      //... Do something ...
} else {
      //... Do something else ...
}
```

```
if (i % 2 == 0) {
      System.out.println (i + “is Even”);
} else {
      System.out.println (i + “is Odd”);
}
```



QAJAVAECL v1.1

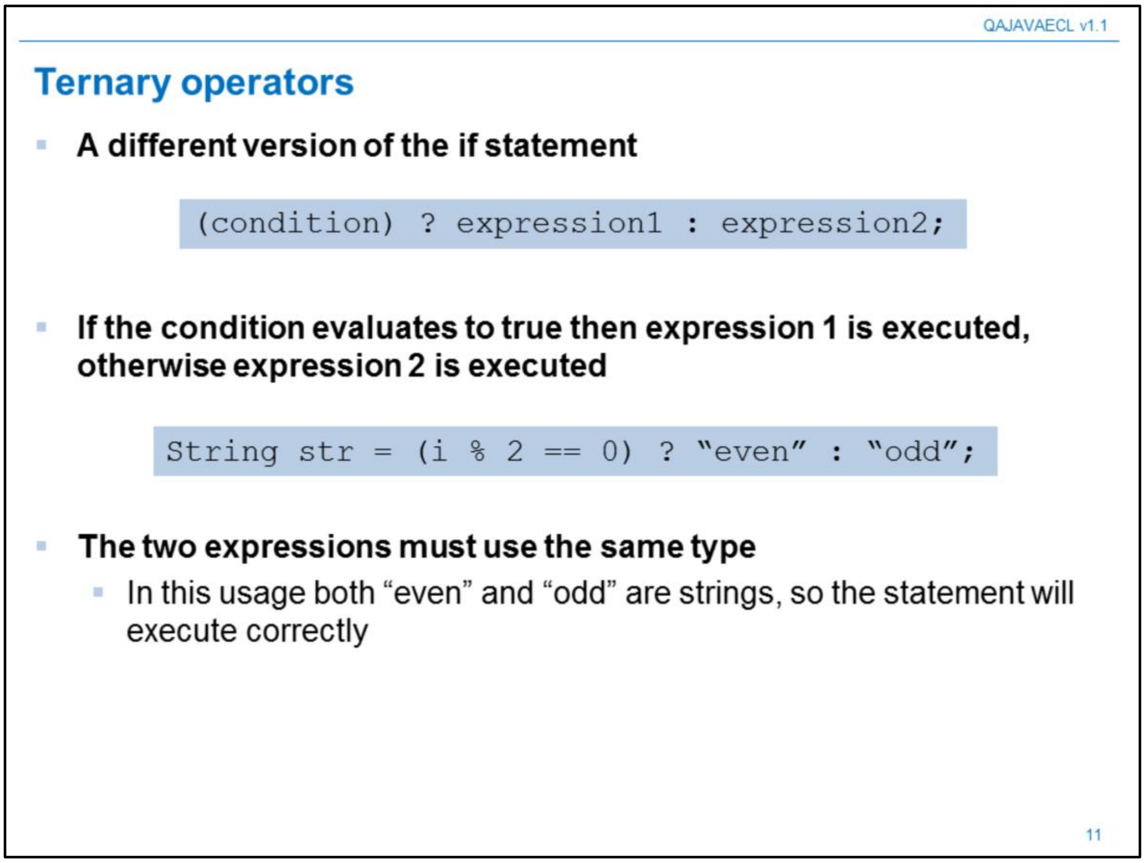
if … else if … else

- **If we want to check for more than one state at a time**
  - We can nest if statements, but using “else if” is safer

```
if (condition) {
      //... Do something ...
} else if (condition2) {
      //... Do something else ...
} else {
      //... An all other cases do this
}
```

```
if (i == 1) {
      System.out.println (“i is 1”);
} else if (i == 2) {
      System.out.println (“i is 2”);
} else {
      System.out.println (“i is not 1 or 2”);
}
```

10



QAJAVAECL v1.1

Ternary operators

- **A different version of the if statement**

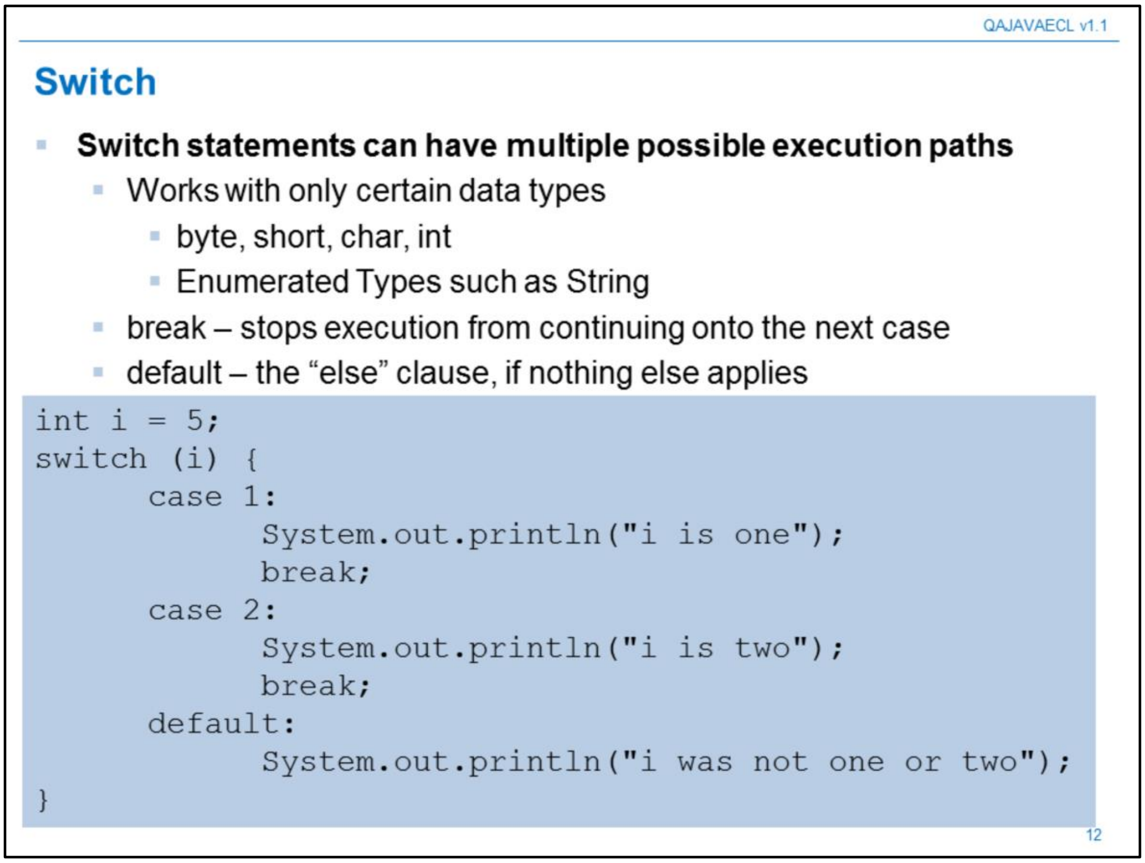
```
(condition) ? expression1 : expression2;
```

- **If the condition evaluates to true then expression 1 is executed, otherwise expression 2 is executed**

```
String str = (i % 2 == 0) ? "even" : "odd";
```

- **The two expressions must use the same type**
  - In this usage both “even” and “odd” are strings, so the statement will execute correctly

11



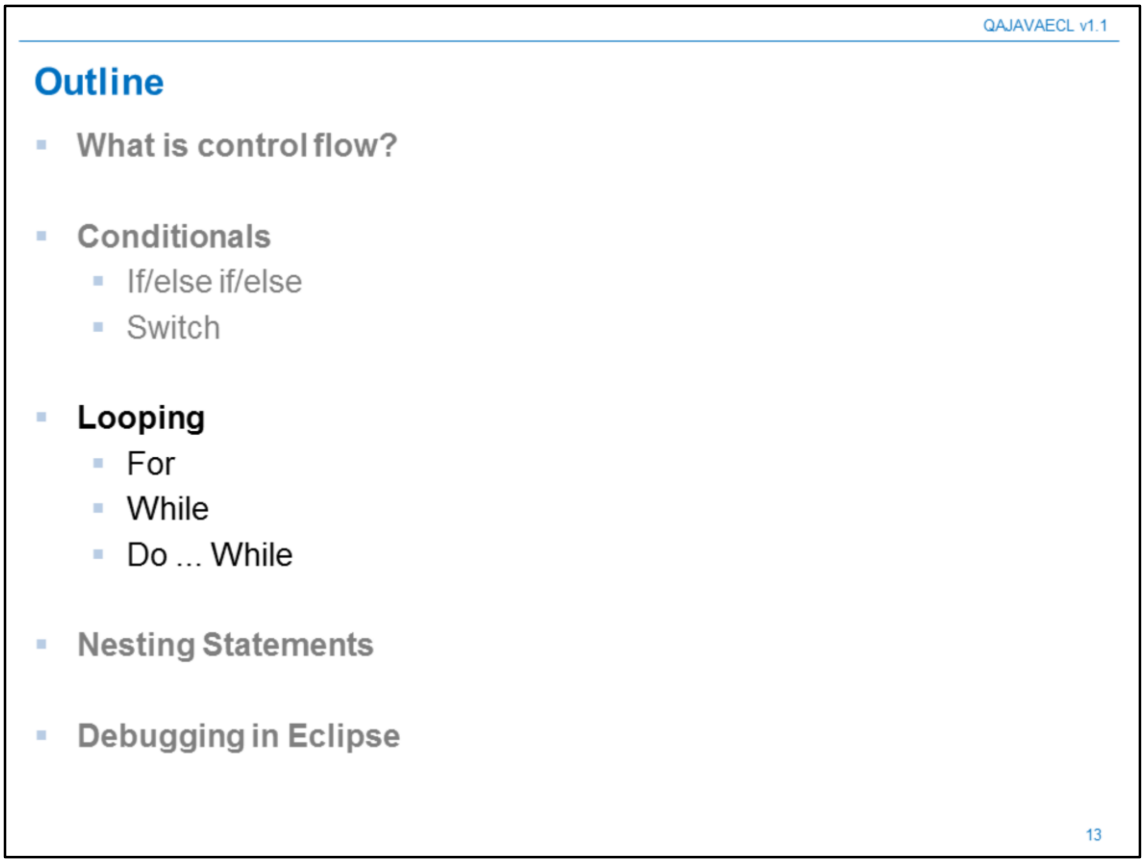
QAJAVAECL v1.1

Switch

- **Switch statements can have multiple possible execution paths**
  - Works with only certain data types
    - byte, short, char, int
    - Enumerated Types such as String
  - break – stops execution from continuing onto the next case
  - default – the “else” clause, if nothing else applies

```
int i = 5;
switch (i) {
      case 1:
            System.out.println("i is one");
            break;
      case 2:
            System.out.println("i is two");
            break;
      default:
            System.out.println("i was not one or two");
}
```

12

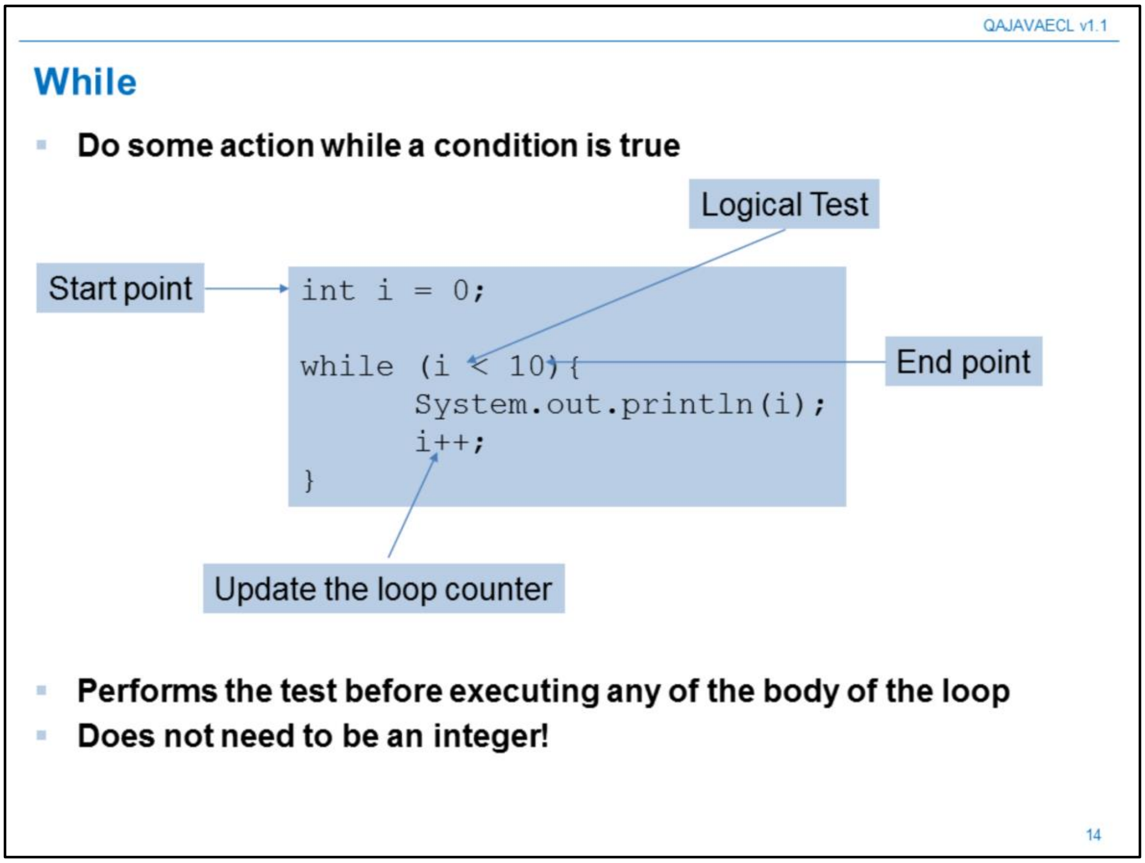


QAJAVAECL v1.1

Outline

- What is control flow?
- Conditionals
  - If/else if/else
  - Switch
- **Looping**
  - For
  - While
  - Do ... While
- Nesting Statements
- Debugging in Eclipse

13



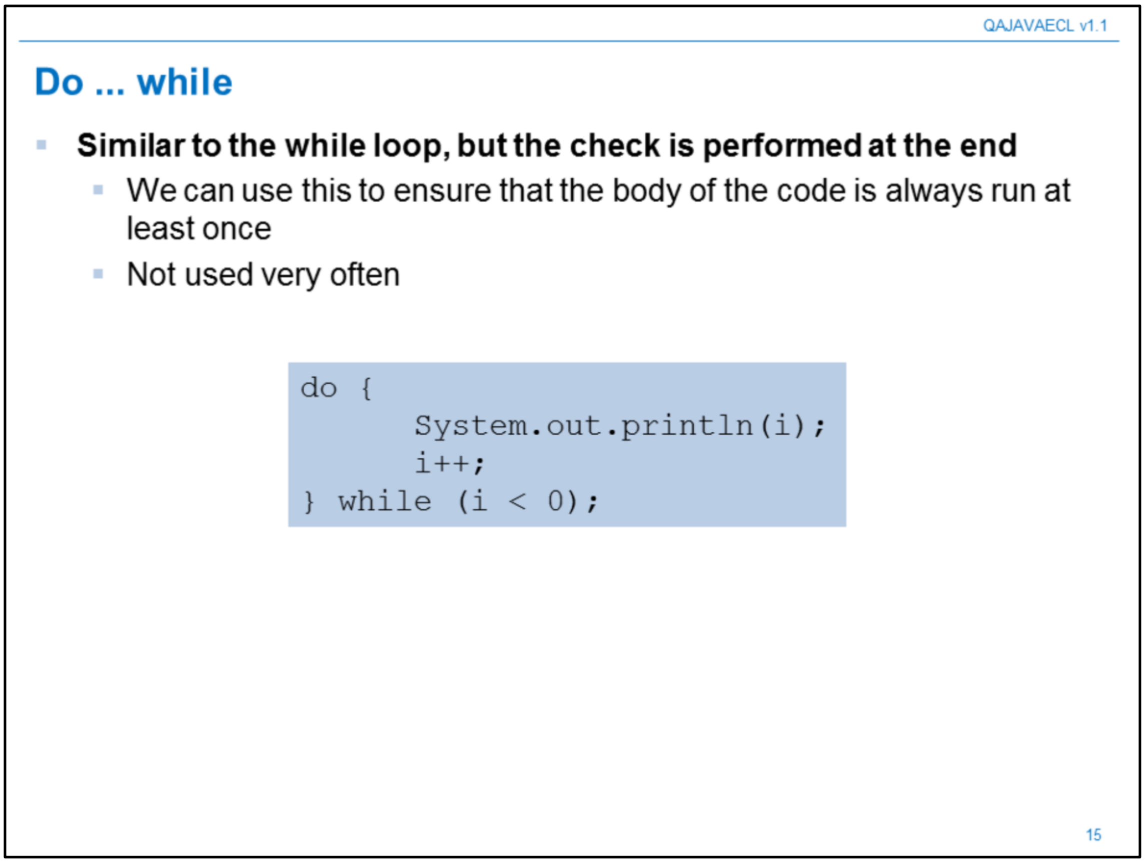
While

- Do some action while a condition is true

```
int i = 0;

while (i < 10){
      System.out.println(i);
      i++;
}
```

Logical Test
Start point
End point
Update the loop counter

- Performs the test before executing any of the body of the loop
- Does not need to be an integer!



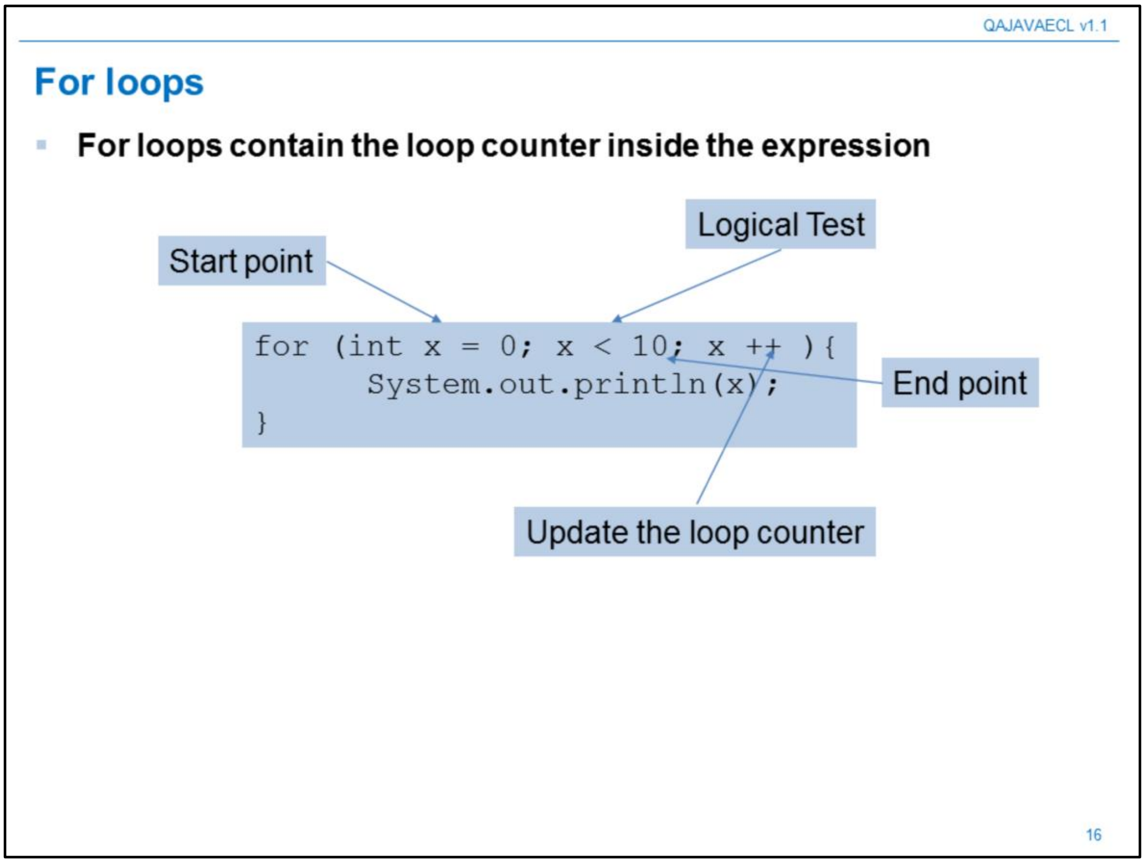
QAJAVAECL v1.1

Do ... while

- **Similar to the while loop, but the check is performed at the end**
  - We can use this to ensure that the body of the code is always run at least once
  - Not used very often

```
do {
      System.out.println(i);
      i++;
} while (i < 0);
```

15

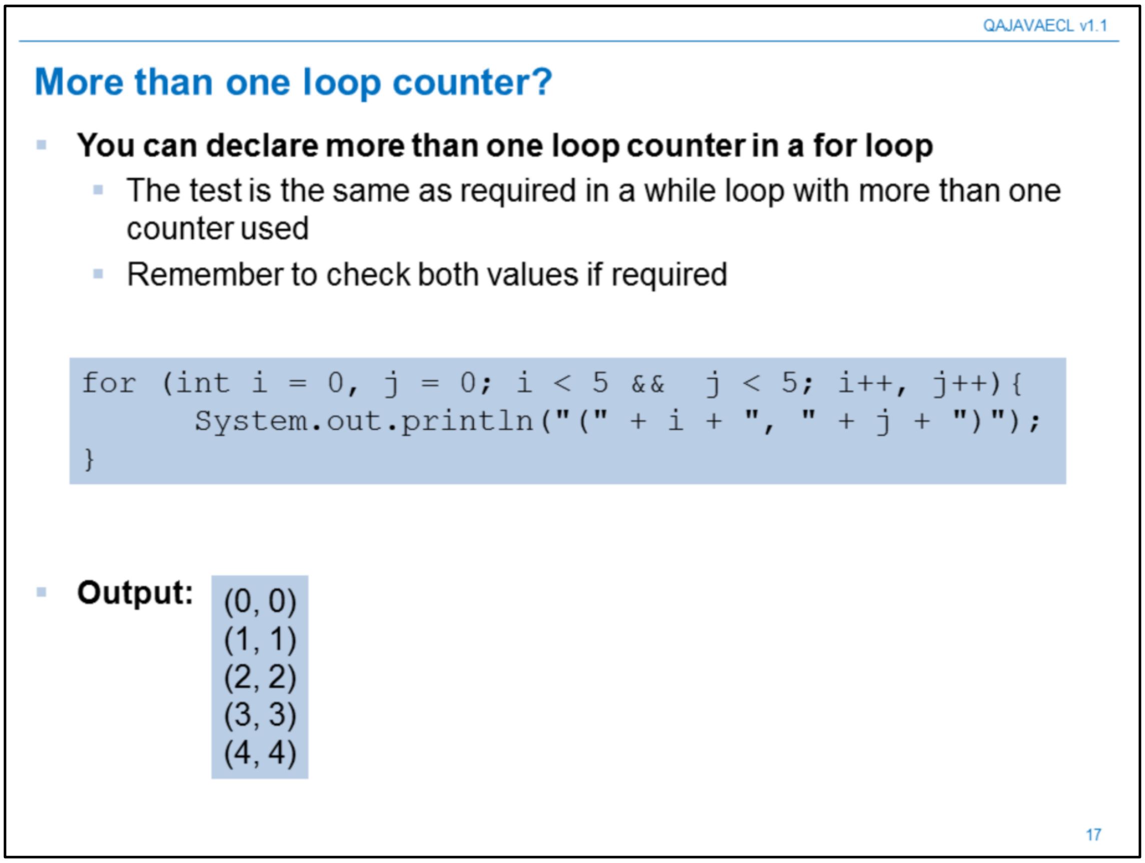


For loops

- For loops contain the loop counter inside the expression

```
for (int x = 0; x < 10; x ++ ){
        System.out.println(x);
}
```

Start point
Logical Test
End point
Update the loop counter



More than one loop counter?

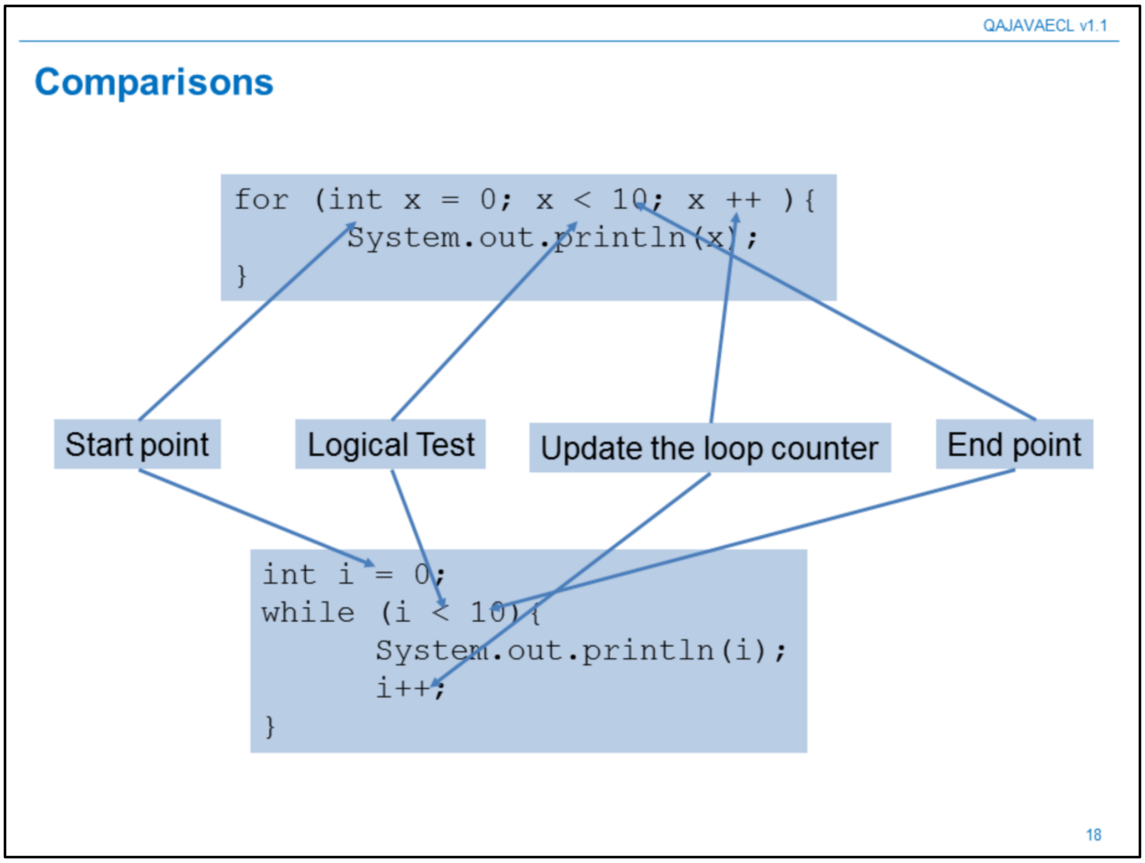
- **You can declare more than one loop counter in a for loop**
  - The test is the same as required in a while loop with more than one counter used
  - Remember to check both values if required

```
for (int i = 0, j = 0; i < 5 &&  j < 5; i++, j++){
      System.out.println("(" + i + ", " + j + ")");
}
```

- **Output:**

```
(0, 0)
(1, 1)
(2, 2)
(3, 3)
(4, 4)
```

17



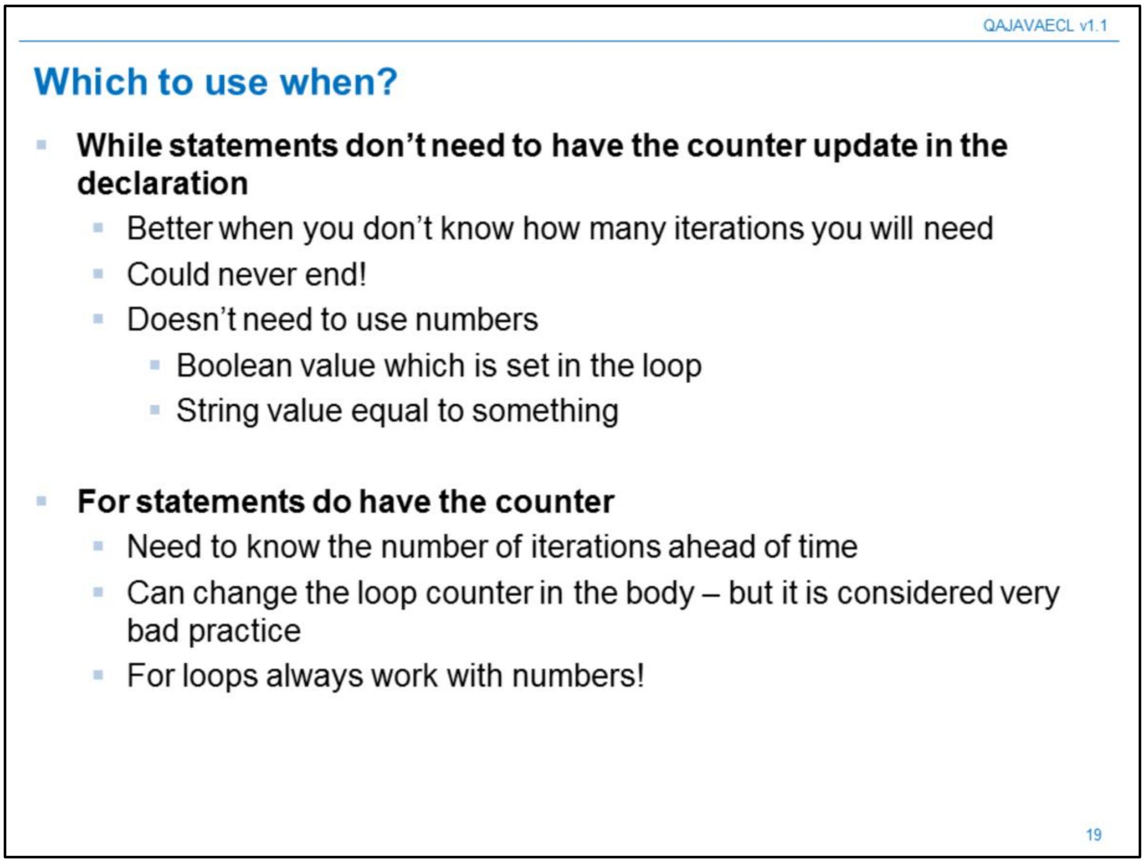
QAJAVAECL v1.1

Comparisons

```
for (int x = 0; x < 10; x ++ ){
      System.out.println(x);
}
```

```
int i = 0;
while (i < 10){
      System.out.println(i);
      i++;
}
```

18



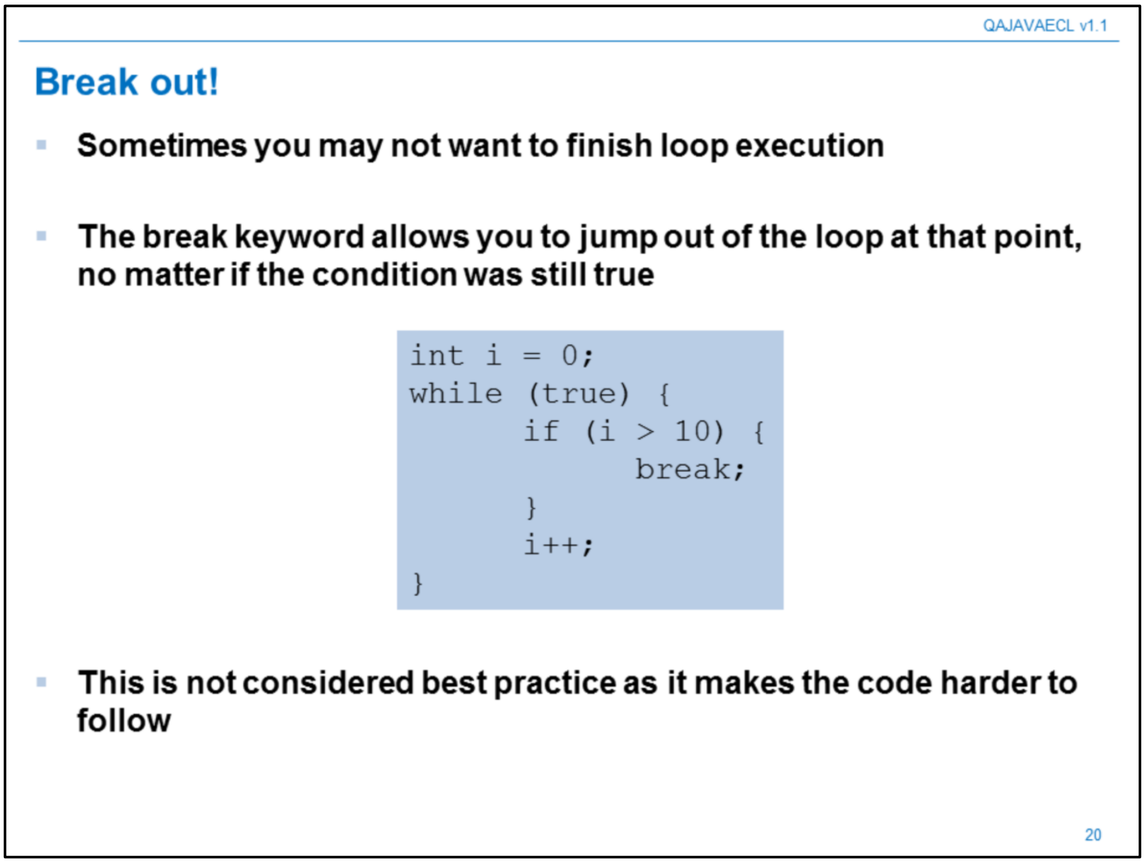
QAJAVAECL v1.1

Which to use when?

- **While statements don't need to have the counter update in the declaration**
  - Better when you don't know how many iterations you will need
  - Could never end!
  - Doesn't need to use numbers
    - Boolean value which is set in the loop
    - String value equal to something

- **For statements do have the counter**
  - Need to know the number of iterations ahead of time
  - Can change the loop counter in the body – but it is considered very bad practice
  - For loops always work with numbers!

19



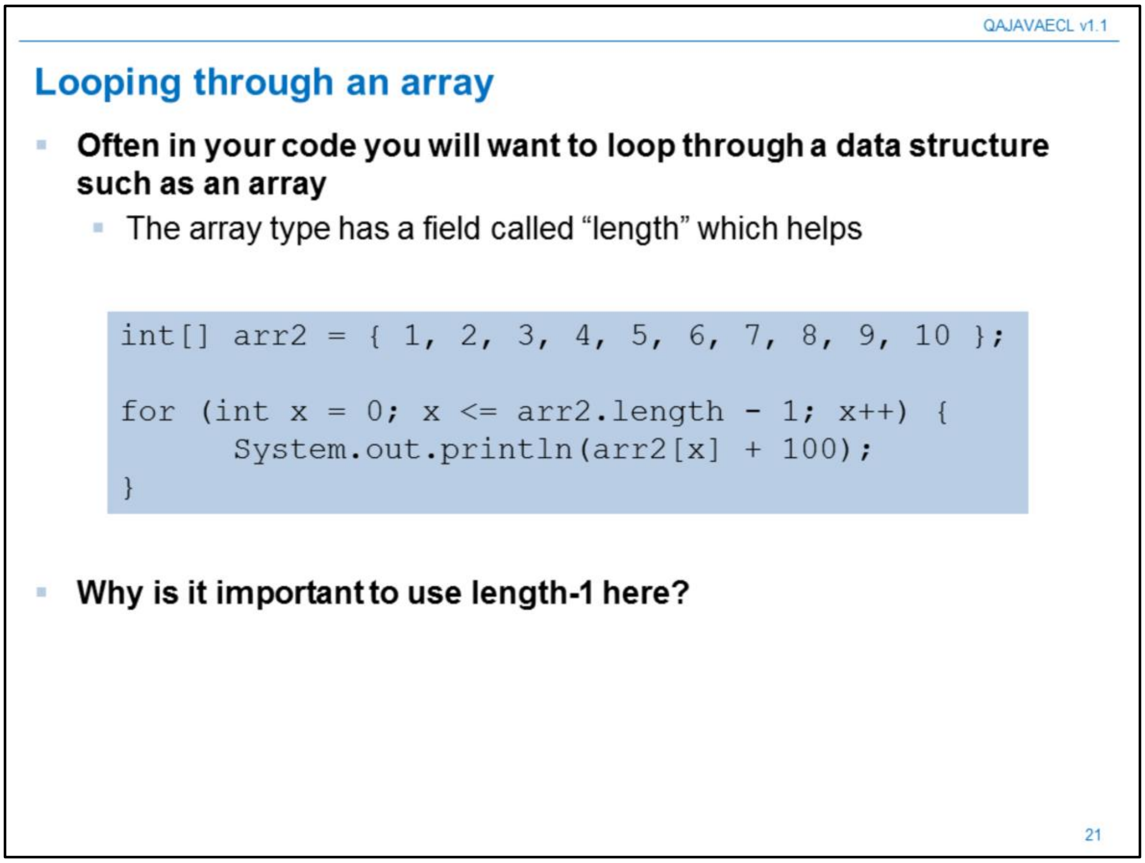
Break out!

- Sometimes you may not want to finish loop execution
- The break keyword allows you to jump out of the loop at that point, no matter if the condition was still true

```
int i = 0;
while (true) {
      if (i > 10) {
            break;
      }
      i++;
}
```

- This is not considered best practice as it makes the code harder to follow

20



QAJAVAECL v1.1

Looping through an array

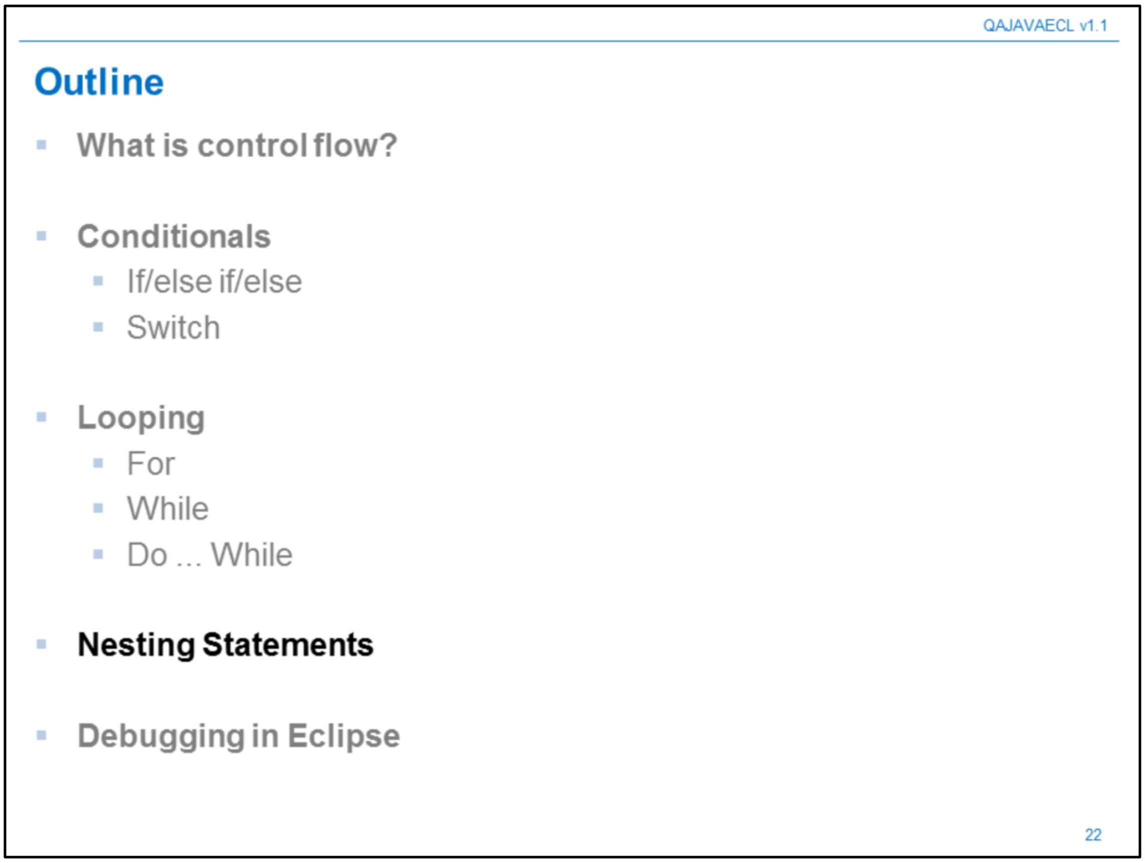
- **Often in your code you will want to loop through a data structure such as an array**
  - The array type has a field called “length” which helps

```
int[] arr2 = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

for (int x = 0; x <= arr2.length - 1; x++) {
      System.out.println(arr2[x] + 100);
}
```

- **Why is it important to use length-1 here?**

21

Indexing an array starts from zero, the length reports how many elements are in the array. The length will always be one higher than the indexing goes to. This is called an off-by-one error

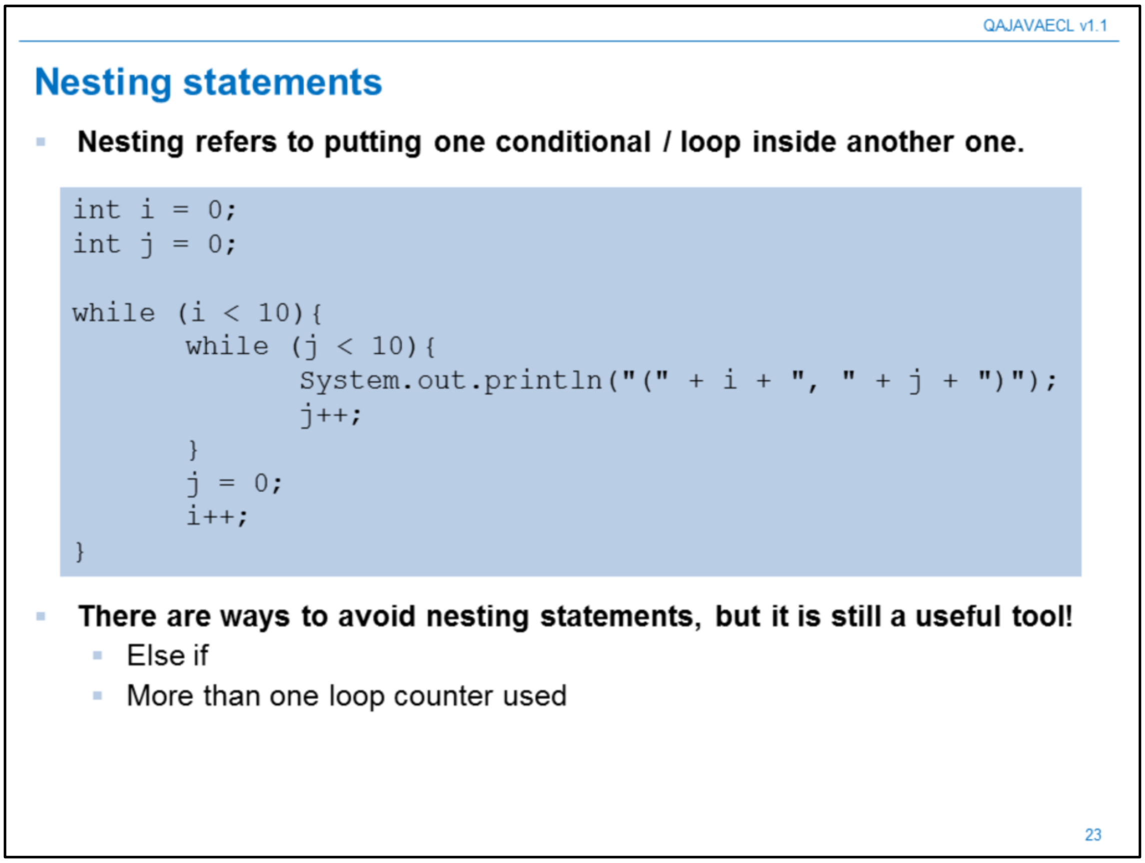


QAJAVAECL v1.1

Outline

- What is control flow?
- Conditionals
  - If/else if/else
  - Switch
- Looping
  - For
  - While
  - Do ... While
- **Nesting Statements**
- Debugging in Eclipse

22



QAJAVAECL v1.1

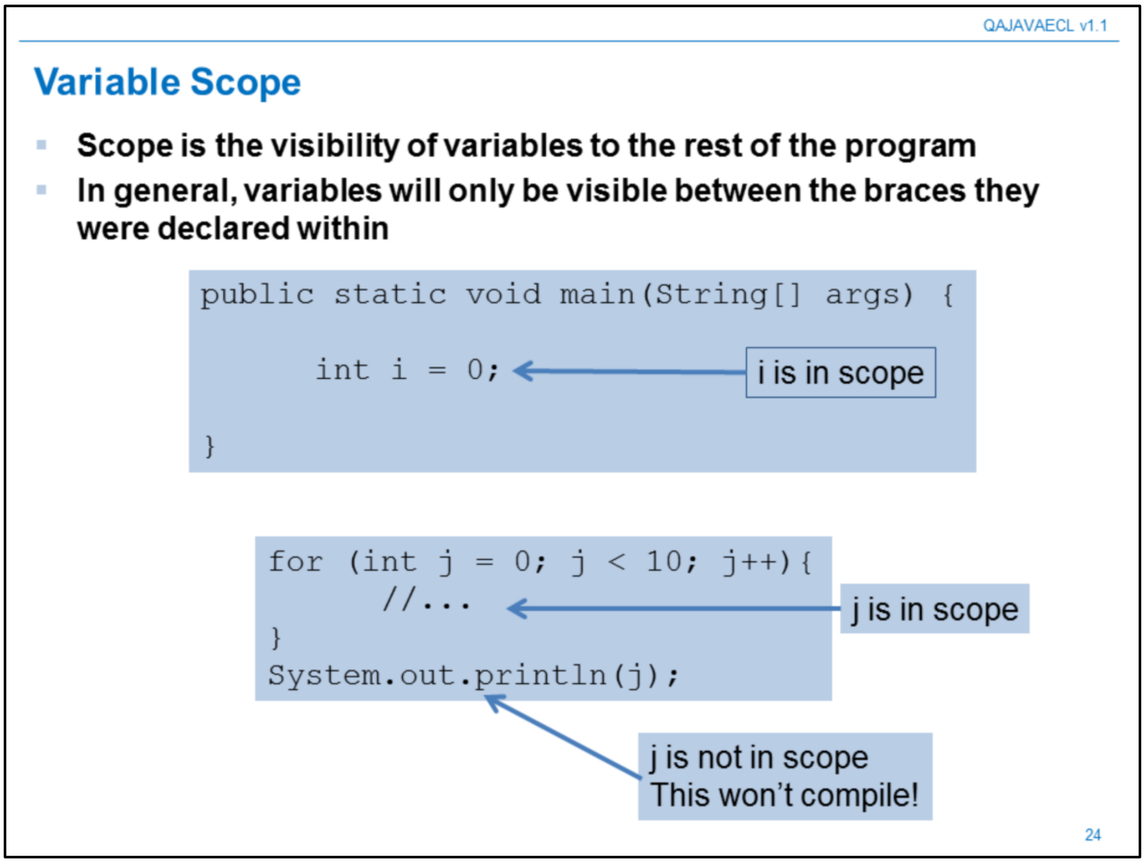
Nesting statements

- **Nesting refers to putting one conditional / loop inside another one.**

```
int i = 0;
int j = 0;

while (i < 10){
        while (j < 10){
                System.out.println("(" + i + ", " + j + ")");
                j++;
        }
        j = 0;
        i++;
}
```

- **There are ways to avoid nesting statements, but it is still a useful tool!**
  - Else if
  - More than one loop counter used

23



QAJAVAECL v1.1

Variable Scope

- Scope is the visibility of variables to the rest of the program
- In general, variables will only be visible between the braces they were declared within

```
public static void main(String[] args) {

      int i = 0;

}
```

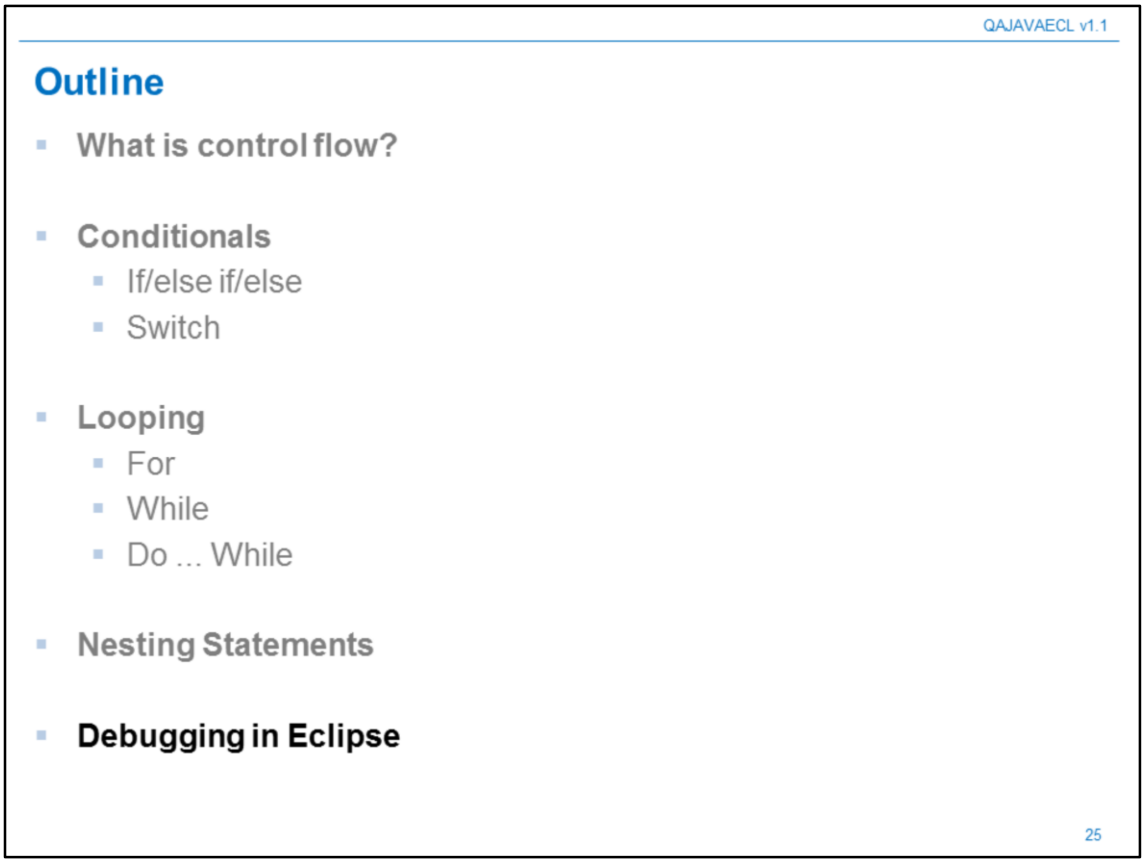
i is in scope

```
for (int j = 0; j < 10; j++){
      //...
}
System.out.println(j);
```

j is in scope

j is not in scope
This won't compile!

24

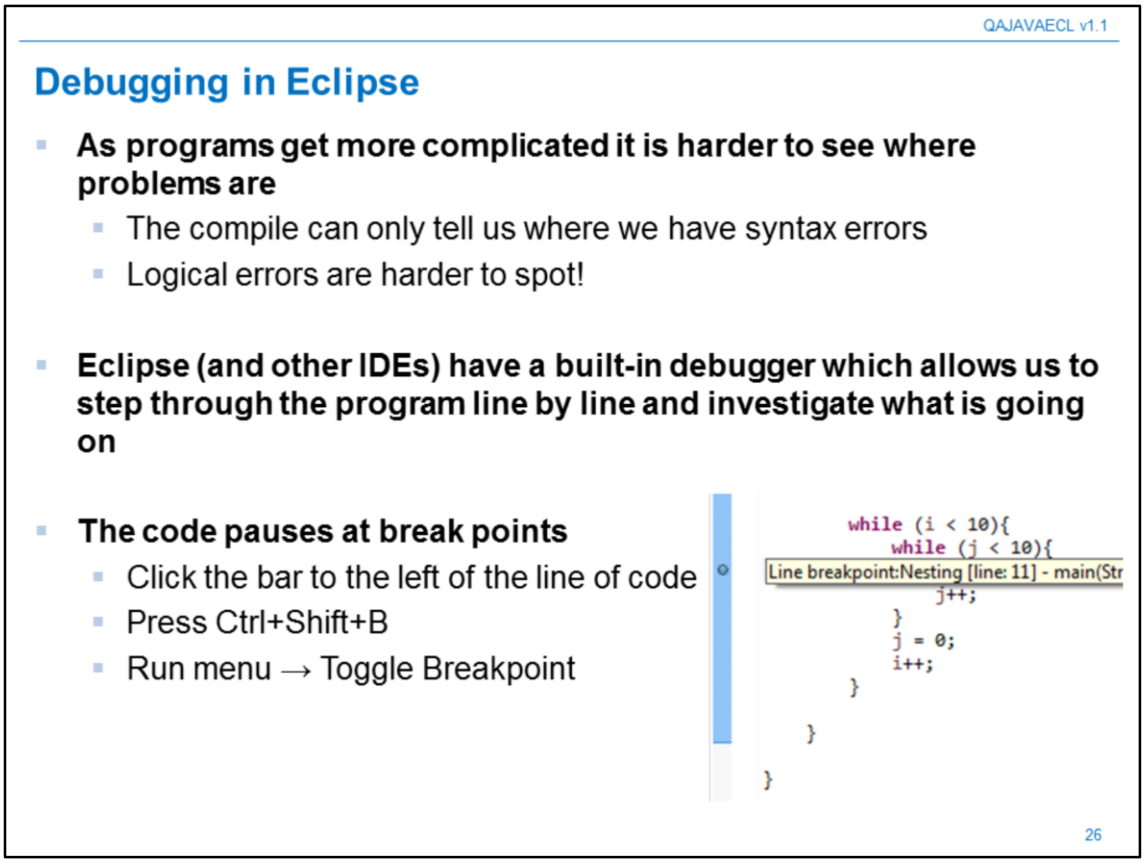


QAJAVAECL v1.1

Outline

- What is control flow?
- Conditionals
  - If/else if/else
  - Switch
- Looping
  - For
  - While
  - Do ... While
- Nesting Statements
- **Debugging in Eclipse**

25



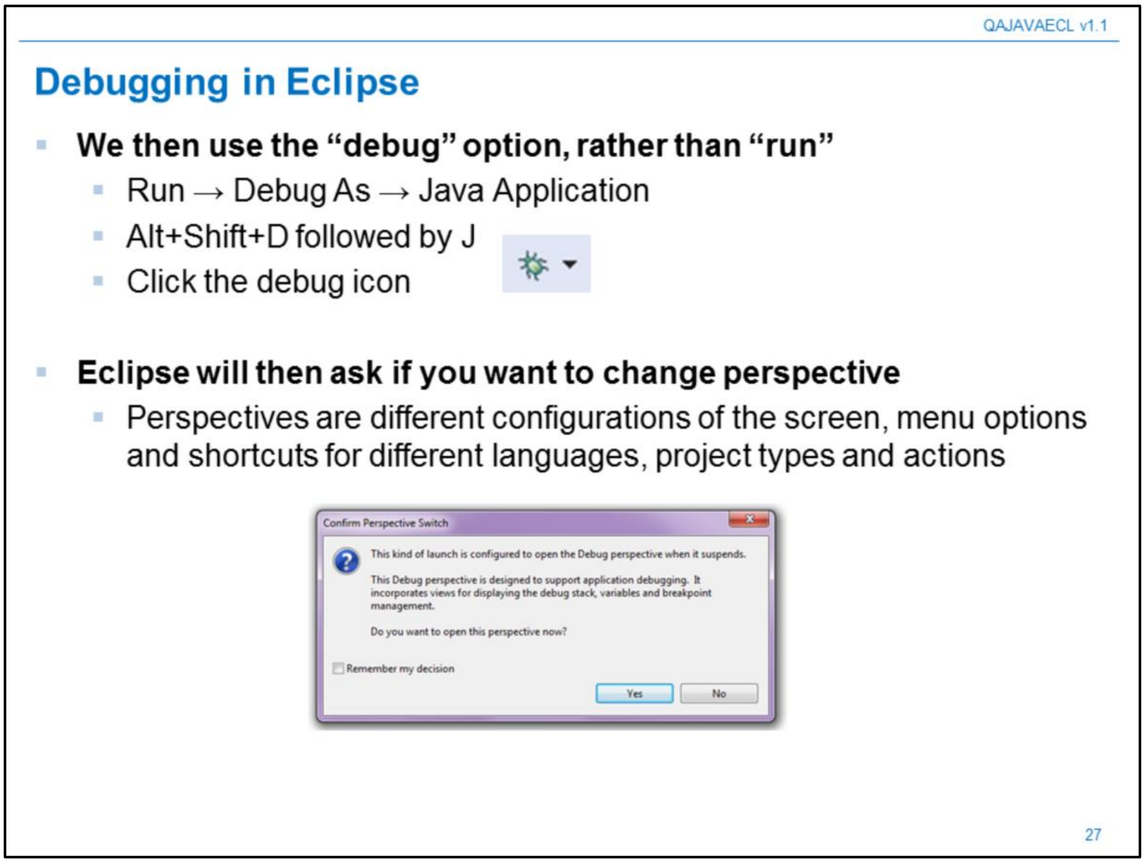
QAJAVAECL v1.1

Debugging in Eclipse

- **As programs get more complicated it is harder to see where problems are**
  - The compile can only tell us where we have syntax errors
  - Logical errors are harder to spot!
- **Eclipse (and other IDEs) have a built-in debugger which allows us to step through the program line by line and investigate what is going on**
- **The code pauses at break points**
  - Click the bar to the left of the line of code
  - Press Ctrl+Shift+B
  - Run menu → Toggle Breakpoint

```
            while (i < 10){
                while (j < 10){
Line breakpoint:Nesting [line: 11] - main(Str
                    j++;
                }
                j = 0;
                i++;
            }

        }

    }
```

26



QAJAVAECL v1.1

Debugging in Eclipse

- **We then use the "debug" option, rather than "run"**
  - Run → Debug As → Java Application
  - Alt+Shift+D followed by J
  - Click the debug icon
- **Eclipse will then ask if you want to change perspective**
  - Perspectives are different configurations of the screen, menu options and shortcuts for different languages, project types and actions

Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

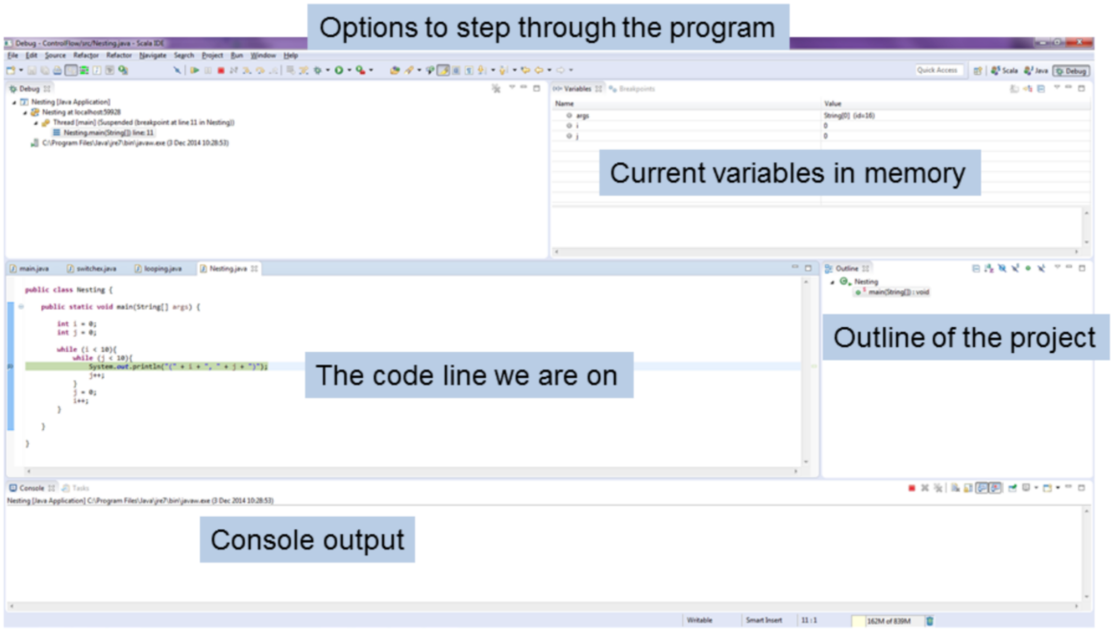
Do you want to open this perspective now?

☐ Remember my decision

Yes No

27

The debug perspective



```
Nesting [Java Application]
  Nesting at localhost:59928
    Thread [main] (Suspended (breakpoint at line 11 in Nesting))
      Nesting.main(String[]) line: 11
  C:\Program Files\Java\jre7\bin\javaw.exe (3 Dec 2014 10:28:53)
```

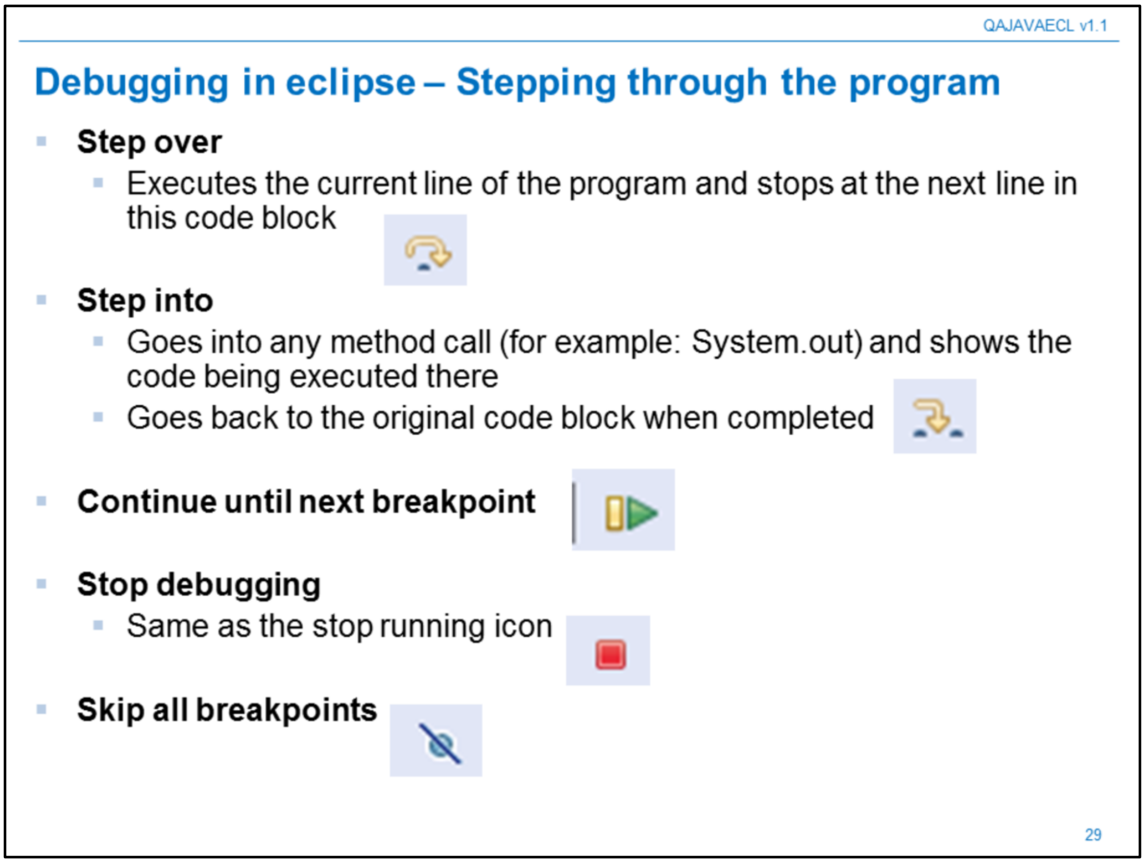
Name	Value
args	String[0] (id=16)
i	0
j	0

```
Nesting
  main(String[]) : void
```

```java
public class Nesting {

    public static void main(String[] args) {

        int i = 0;
        int j = 0;

        while (i < 10){
            while (j < 10){
                System.out.println("(" + i + ", " + j + ")");
                j++;
            }
            j = 0;
            i++;
        }

    }

}
```

```
Nesting [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (3 Dec 2014 10:28:53)
```

28



QAJAVAECL v1.1

Debugging in eclipse – Stepping through the program

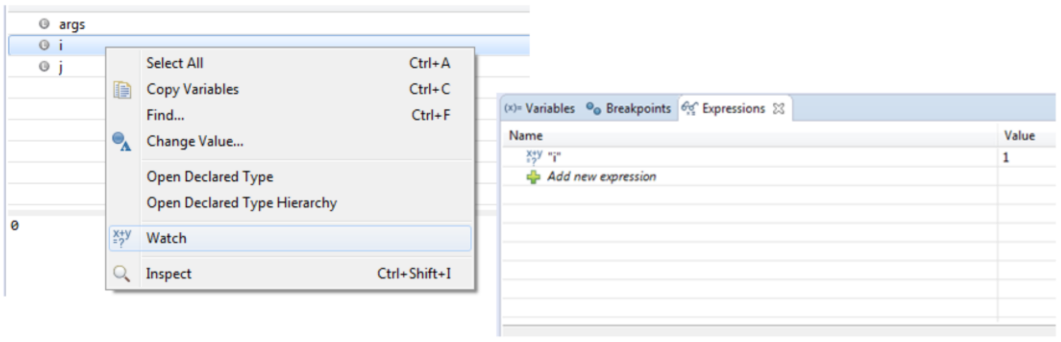
- **Step over**
  - Executes the current line of the program and stops at the next line in this code block
- **Step into**
  - Goes into any method call (for example: System.out) and shows the code being executed there
  - Goes back to the original code block when completed
- **Continue until next breakpoint**
- **Stop debugging**
  - Same as the stop running icon
- **Skip all breakpoints**

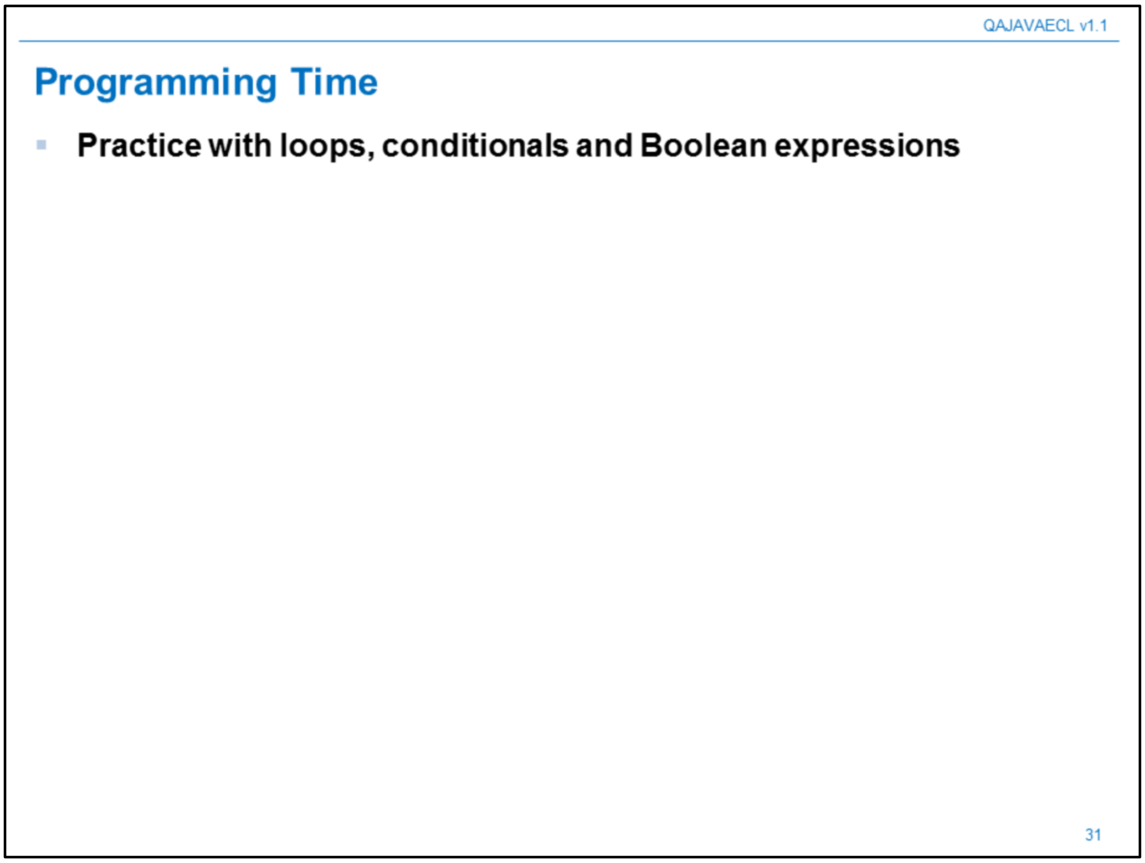
29

Debugging in eclipse – Watching variables

- **In a program with a lot of variables it can be difficult to find the specific value we want to watch**
  - We can chose individual variables and specify that we are interested in these no matter where we are in the program

- **Right-click the variable we are interested in and select "watch"**



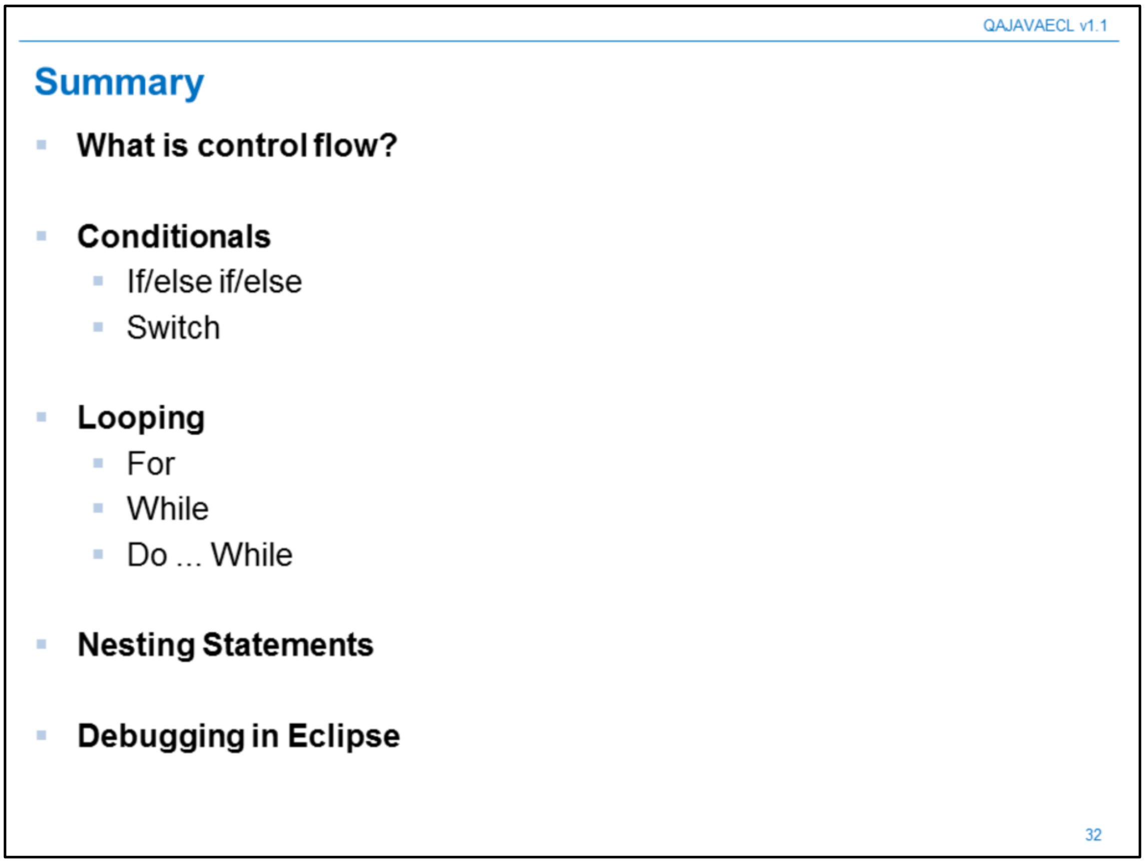


QAJAVAECL v1.1

Programming Time

- Practice with loops, conditionals and Boolean expressions

31



QAJAVAECL v1.1

Summary

- **What is control flow?**
- **Conditionals**
  - If/else if/else
  - Switch
- **Looping**
  - For
  - While
  - Do ... While
- **Nesting Statements**
- **Debugging in Eclipse**

32