4

Control Flow Statements

Objectives

After completing this lesson, you should be able to do the following:

- Use decision-making constructs
- Perform loop operations
- Write `switch` statements
- Arrays Loops
- Errors and Debugging

Course Roadmap

Fundamentals of Programming

- Lesson 1: Introduction to Fundamentals of Programming
- Lesson 2: How Programming Languages Work
- Lesson 3: Data and Types
- **Lesson 4: Control Flow Statement Errors & Debugging** — **You are here!**
- Lesson 5: : Structured Modular & Object Oriented

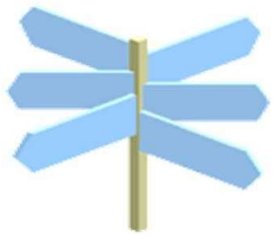
Basic Flow Control Types

Flow control can be categorized into four types:

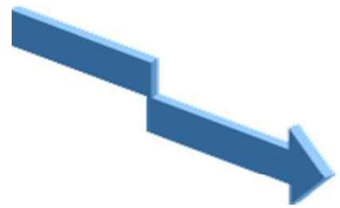
Sequential

Iteration

Selection



Transfer



Using Flow Control

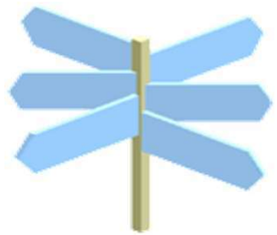
- Each simple statement terminates with a semicolon (;).
- Group statements by using braces { }.
- Each block executes as a single statement in the flow of control structure.

```
{
  boolean finished = true;
  System.out.println("i = " + i);
  i++;
}
```

`if` Statement

General:

```
if ( boolean_expr )
    statement1;
[else
    statement2;]
```



Examples:

```
if (i % 2 == 0)
   System.out.println("Even");
else
   System.out.println("Odd");
…
```

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

Nested `if` Statements

```
if (speed >= 25)
  if (speed > 65)
      System.out.println("Speed over 65");
  else
      System.out.println("Speed >= 25 but <= 65");
  else
    System.out.println("Speed under 25");
```

```
if (speed > 65)
  System.out.println("Speed over 65");
else if (speed >= 25)
        System.out.println("Speed greater… to 65");
     else
        System.out.println("Speed under 25");
```

Guided Practice: Spot the Mistakes

1

```
int x = 3, y = 5;
if (x >= 0)
  if (y < x)
    System.out.println("y is less than x");
else
    System.out.println("x is negative");
```

2

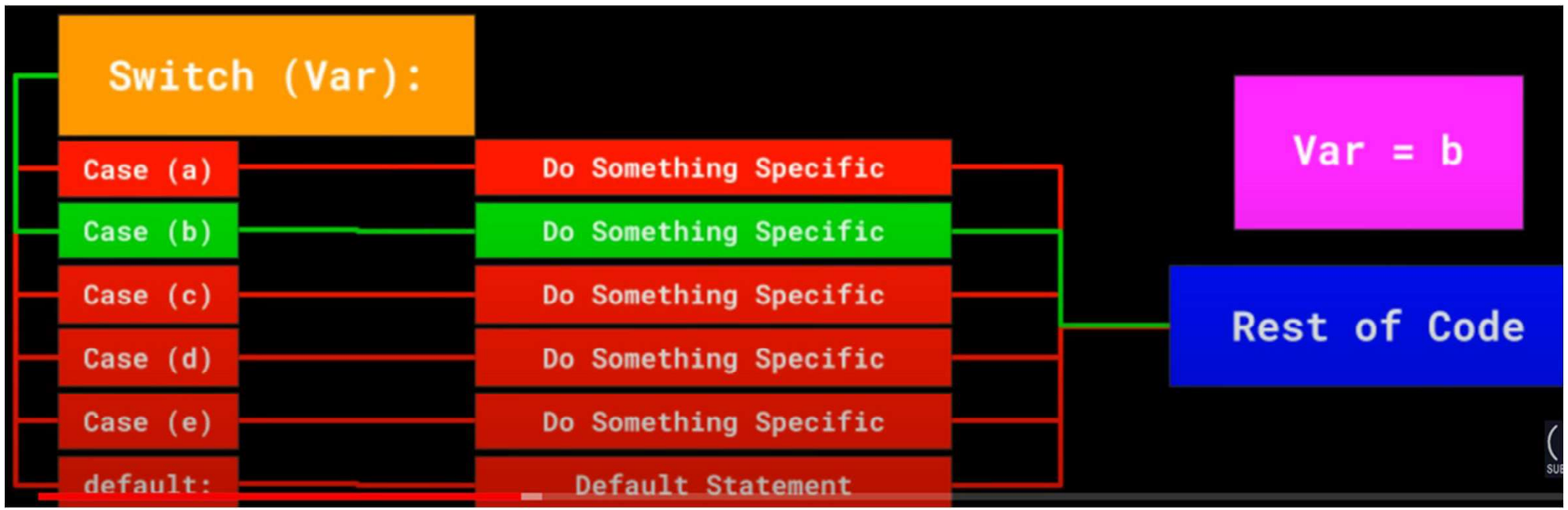
```
int x = 7;
if (x = 0)
    System.out.println("x is zero");
```

3

```
int x = 14, y = 24;
if ( x % 2 == 0  &&  y % 2 == 0 );
    System.out.println("x and y are even");
```

What are Conditional Statements? Switch statement

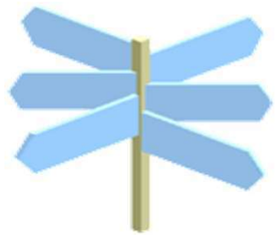
- **Start with a colon (:)**



`switch` Statement

```
switch ( integer_expr ) {

    case constant_expr1:
        statement1;
        break;
    case constant_expr2:
        statement2;
        break;
    [default:
        statement3;]
}
```



- The `switch` statement is useful when selecting an action from several alternative integer values.
- `Integer_expr` must be `byte`, `int`, `char`, `short` or **`String`**

More About the `switch` Statement

- `case` labels must be constants.
- Use `break` to jump out of a switch.
- You should always provide a default.

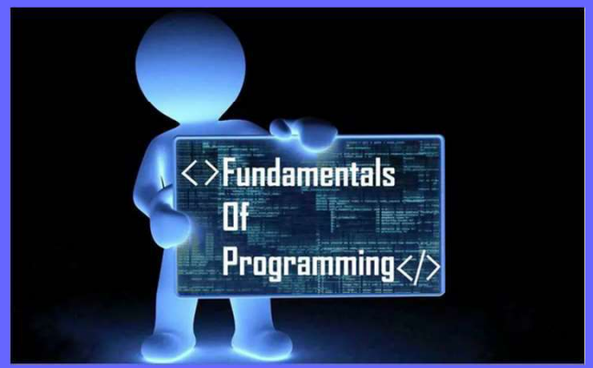
```
switch (choice) {
  case 37:
     System.out.println("Coffee?");
     break;

  case 45:
     System.out.println("Tea?");
     break;

  default:
     System.out.println("???");
     break;
}
```

What are Conditional Statements? The usefulness of Conditional Statements

- Add Variability to programming
  - Program runs differently based on user input
- If a user does something, we want to be able to adapt accordingly without, a program would run the same way every time



Arrays

What are Arrays? Where do Variables Fail?

- Variables are very good at storing singular bits of information
- As a result, they are unable to hold more than one piece of data

What are Arrays? grocery list example

What are Arrays? well what are they

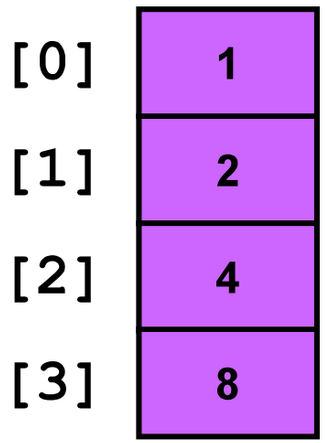
- An array is a list of something
  - can be integers
  - can be strings
  - can be any data type
- All information in an array is related
- Like columns in google sheet or excel

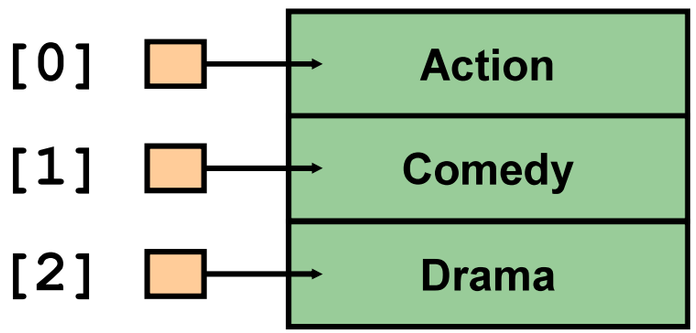
What are Arrays? company example

Arrays

An array is a collection of variables of the same type.

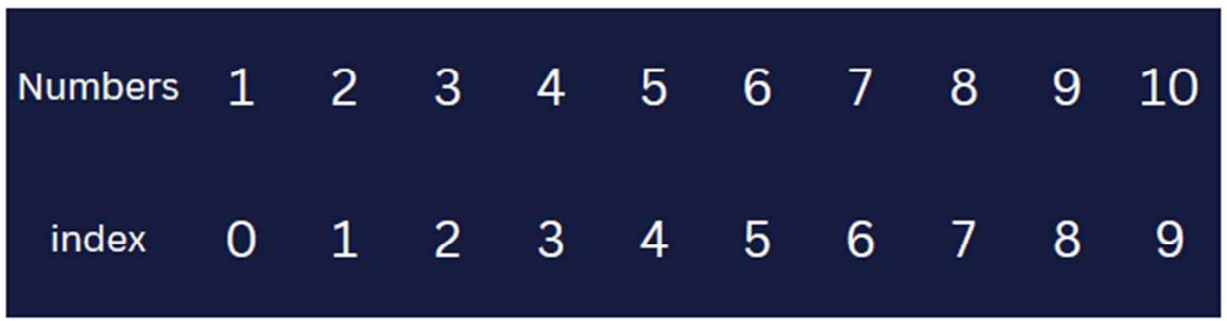
- Each element can hold a single item.
- Items can be primitives or object references.
- The length of the array is fixed when it is created.





What are Arrays? Referencing arrays

- The single most important thing to note about arrays is how we reference each element inside of them
  - In programming we use indexes, that numbers place in an array



What are Arrays? Creating Arrays

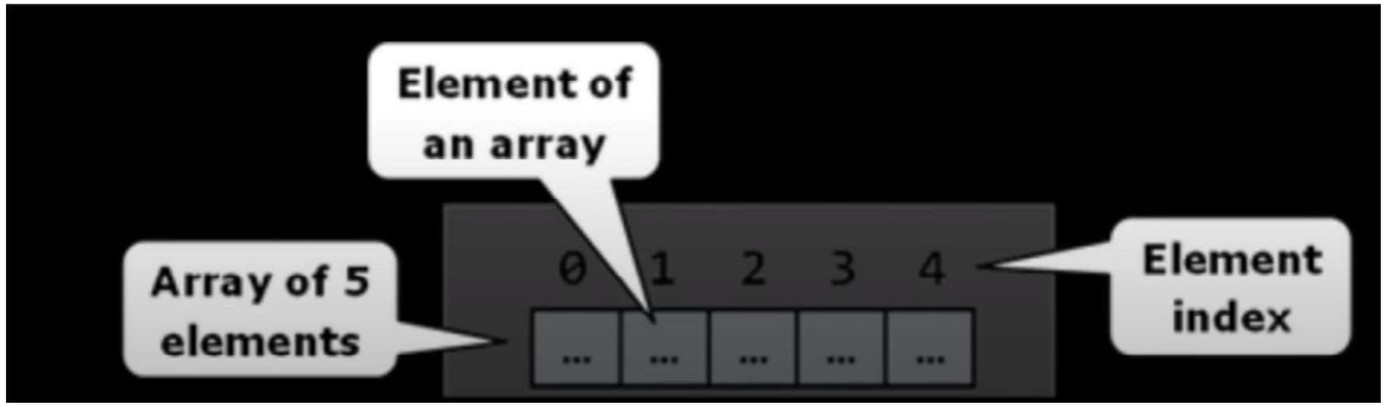
- Populate first
  - Insert the elements you would like in the array immediately
  - String grocerList[] = {"egg","milk","sugar","oil","ghee"}

What are Arrays? Creating Arrays

- Populate later
- Create an array with specific size and choose to add the elements later
  - String grocerList[5];
  - groceryList[0] = "egg";
  - groceryList[1] = "milk";
  - groceryList[2] = "sugar";
  - groceryList[3] = "oil";
  - groceryList[4] = "ghee";

What are Arrays? Array Sizes

- When we create array their sizes are FINAL cannot be incremented or decremented in size through conventional methods
- This is what allows us to easily access their index



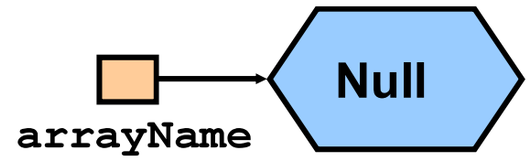
What are Arrays? Array Types

- When initializing the array , you must determine its type then and there
- Example (String array, integer array, double array etc.)
- They have to all be of the same type [Homogeneous]

Creating an Array of Primitives

1. Declare the array.

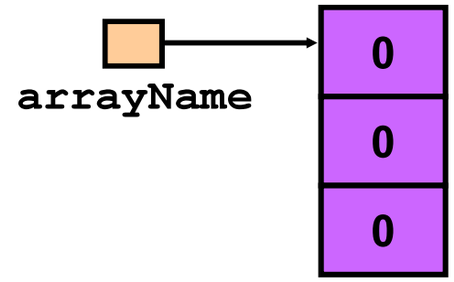
```
type[] arrayName;
… or …
type arrayName[];
```



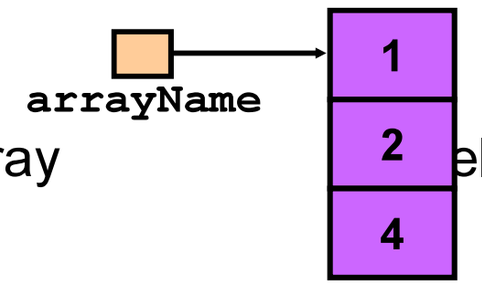
`type` is a primitive, such as `int` and so on.

2. Create the array object.

```
// Create array object syntax
arrayName = new type[size];
```



3. Initialize the array elements (optional).

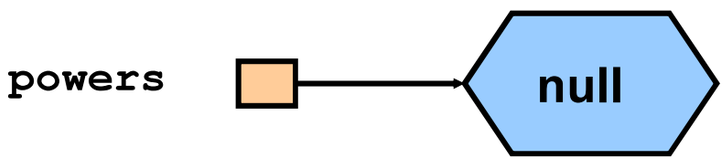


Declaring an Array

- Create a variable to reference the array object:

```
int[] powers;  // Example
```

- When an array variable is declared:
  - Its instance variable is initialized to `null` until the array object has been created



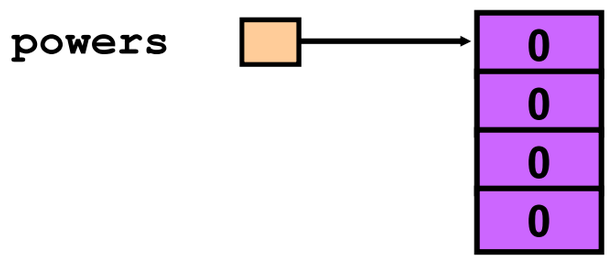
  - Its method variable is unknown until the object is created

Creating an Array Object

- Create an array of the required length and assign it to the array variable:

```
int[] powers;          // Declare array variable
powers = new int[4];   //Create array object
```

  - Create the array object by using the `new` operator.
- The contents of an array of primitives are initialized automatically.

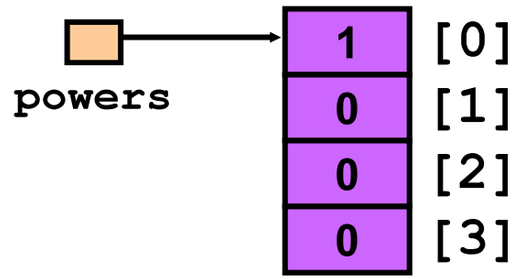


Initializing Array Elements

- Assign values to individual elements:

```
arrayName[index] = value;
```

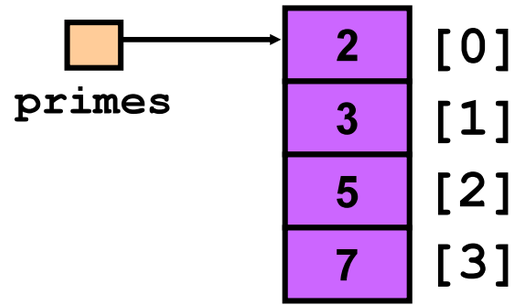
```
powers[0] = 1;
```



- Create and initialize arrays at the same time:

```
type[] arrayName = {valueList};
```

```
int[] primes = {2, 3, 5, 7};
```



What are Arrays? 2D arrays

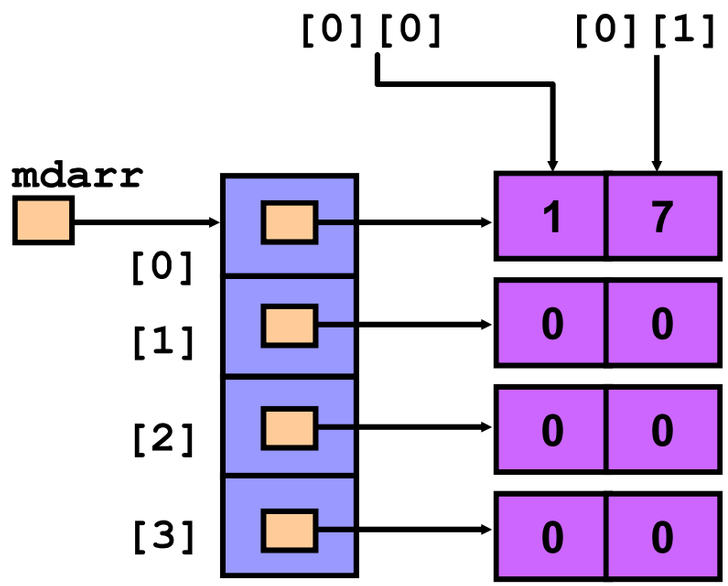
- Putting an array inside an array is known as a 2- dimensional array
- Similar to Matrices in math/physics classes

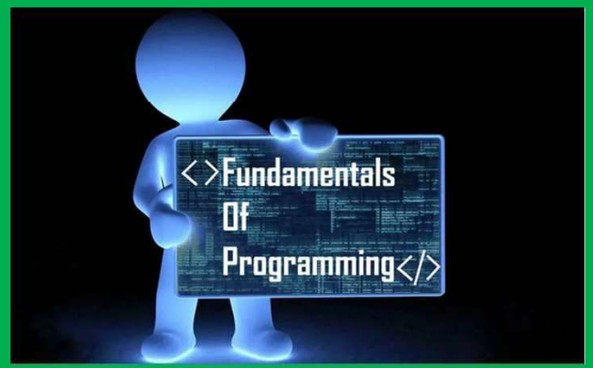
Index	0	1	2	3
0	Ayton	Alex	Arnold	Ashton
1	Bob	Ben	Bryan	Billy
2	Clint	Chris	Colton	Cal

Java supports arrays of arrays:

```
type[][] arrayname = new type[n1][n2];
```

```
int[][] mdarr = new int[4][2];
mdarr[0][0] = 1;
mdarr[0][1] = 7;
```





Loops

What are Loops?

- How can we save have code repeat segments without repeating lines of code?
- A loop is a statement that is used to run certain instructions repeatedly
- Very useful for repeated sections of code

What are Loops? Loop example

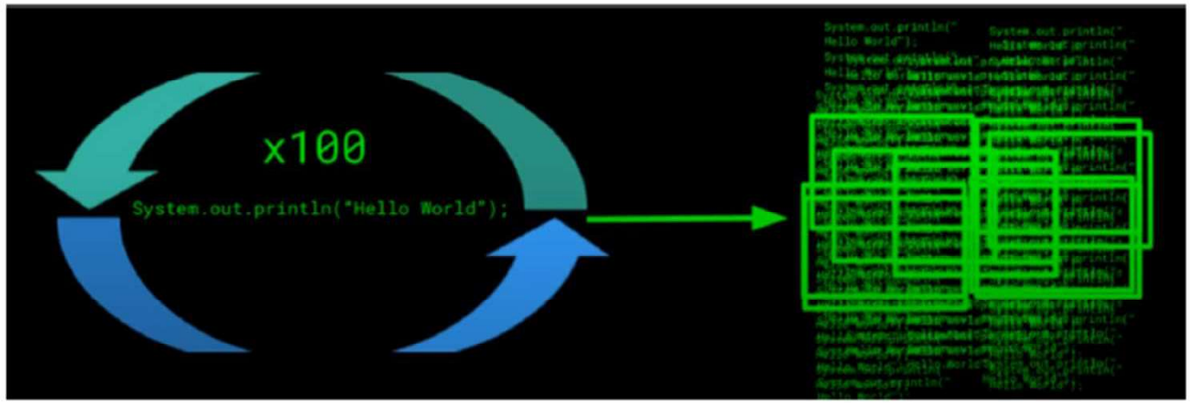
- If we wanted to print something 15 times
- we could use 15 print statements

```
System.out.println("Hello World");
System.out.println("Hello World");
System.out.println("Hello World");
System.out.println("Hello World");
System.out.println("Hello World");
System.out.println("Hello World");
System.out.println("Hello World");
System.out.println("Hello World");
System.out.println("Hello World");
System.out.println("Hello World");
```

```
System.out.println("Hello World")
System.out.println("Hello World")
System.out.println("Hello World")
System.out.println("Hello World")
System.out.println("Hello World")
System.out.println("Hello World")
System.out.println("Hello World")
System.out.println("Hello World")
System.out.println("Hello World")
System.out.println("Hello World")
```

What are Loops? Loop example

- Or we could use a loop
- Allows us to repeat parts of code multiple times



Looping

- There are three types of loops :
  - `while`
  - `do…while`
  - `for`
- All (counter-controlled) loops have four parts:
  - Initialization
  - Body
  - Increment
  - Termination

while Loop

`while` is the simplest loop statement and contains the following general form:

```
while ( boolean_expr )
    statement;
```

Example:

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

do...while Loop

`do...while` loops place the test at the end:

```
do
    statement;
while ( termination );
```

Example:

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

`for` Loop

`for` loops are the most common loops:

```
for ( initialization; termination; increment )
    statement;
```

Example:

```
for (int i = 0; i < 10; i++)
    System.out.println(i);
```

How could this `for` loop be written as a `while` loop?

More About the `for` Loop

- Variables can be declared in the initialization part of a `for` loop:

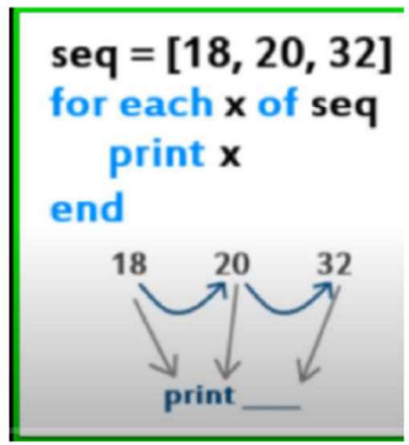
```
for (int i = 0; i < 10; i++)
    System.out.println("i = " + i);
```

- Initialization and increment can consist of a list of comma-separated expressions:

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

What are Loops? For each loop

- The foreach loop
  - used for iterating through entire arrays or lists
- The loop will go through each element in the array and carry out a set of instructions for each value
- Useful for performing operations across an entire collections of data



Arrays and `for-each` Loop

```
1    public class ArrayOperations {
2        public static void main(String args[]){
3
4            String[] names = new String[3];
5
6            names[0] = "Blue Shirt";
7            names[1] = "Red Shirt";
8            names[2] = "Black Shirt";
9
10           int[] numbers = {100, 200, 300};
11
12           for (String name:names){
13               System.out.println("Name: " + name);
14           }
15
16           for (int number:numbers){
17               System.out.println("Number: " + number);
18           }
19       }
20   }
```

Arrays are objects. Array objects have a final field length.

Guided Practice: Spot the Mistakes

1

```
int x = 10;
while (x > 0);
  System.out.println(x--);
System.out.println("We have lift off!");
```

2

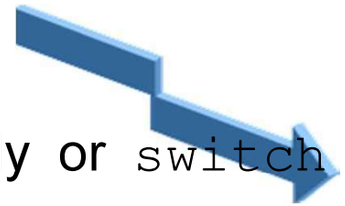
```
int x = 10;
while (x > 0)
  System.out.println("x is " + x);
  x--;
```

3

```
int sum = 0;
for (; i < 10; sum += i++);
System.out.println("Sum is " + sum);
```

`break` Statement

- Breaks out of a loop or `switch` statement
- Transfers control to the first statement after the loop body or `switch` statement
- Can simplify code but must be used sparingly

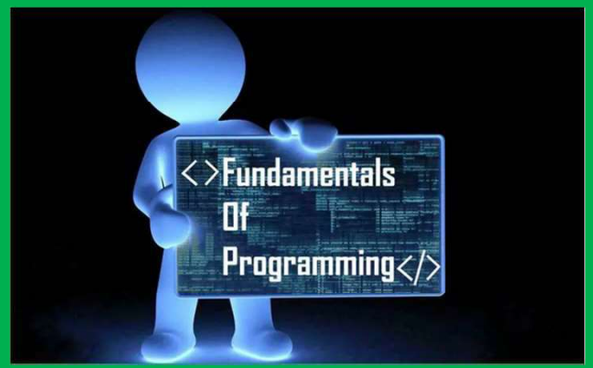


```
…
while (age <= 65) {
    balance = (balance+payment) * (1 + interest);
    if (balance >= 250000)
        break;
    age++;
}
…
```

continue Statement

- Skips the iteration of a loop
- Moves on to the next one

```
…
for (int i=0; i<=10; i++) {
//skips the print statement if i is not even
       if(i % 2 != 0) {
              continue;
       }
//prints the integer "i" followed by a space
System.out.print(i + ' ');
}
…
```



Errors

What are Errors?

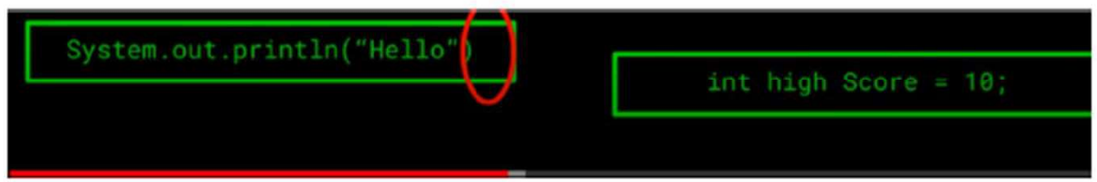
- What happens when our code doesn't work the way we want it too?

What are Errors: Error Introduction

- Code doesn't work as expected
- These are known as errors
  - Come up often in computer science
- Three different types
  - Syntax errors
  - Runtime errors
  - Logic errors

What are Errors: Syntax errors

- Syntax errors
- Parts in your program where you fail to meet the programming rules, resulting in an error
- Usually the easiest of the 3 to solve
- Highlighted by the IDE in most cases



What are Errors: Debugging Syntax errors

- IDEs underline Syntax errors and usually provide helpful hints
- Syntax errors are like small mis spellings or grammatical errors
- Some IDEs will restrict you from running the code unless all syntax errors are cleared

What are Errors: Runtime errors

- Runtime errors
  - Don't appear until you actually "run" the code
- Caused by a part of your code not being able to be computed in a reasonable amount of time
  - Most common form - infinite loop
- Divide By Zero errors

What are Errors: Preventing Runtime errors

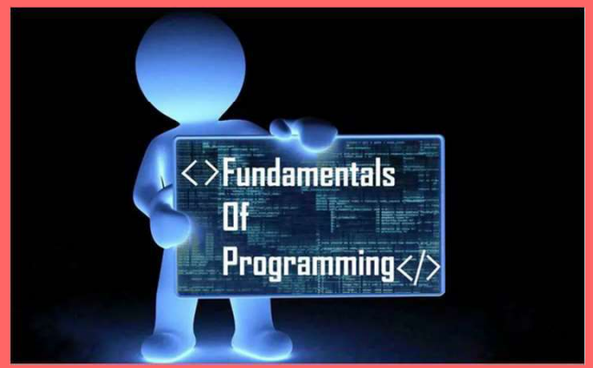
- **Avoid Runtime errors**
  - **Think through the flow of your code before running it (especially loops)**
  - **Carefully plan out your code before writing -> pseudocode**

What are Errors: Logic Errors

- Logic Errors
  - The code runs smoothly without runtime or syntax errors, but the result isn't what you wanted
- Often the hardest type of errors to solve
  - Most of the time, the error is unknown to the programmer

What are Errors: Prevent Logic Errors

- One strategy is coding incrementally
  - Test your application often so that if you mess up you know where the error is



Debugging

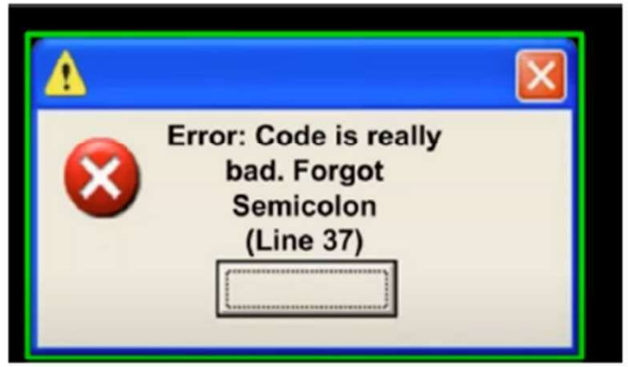
How do we debug the code?

- **How do we fix syntax, runtime, and logic errors in our code?**

How do we debug the code? Debugging Intro

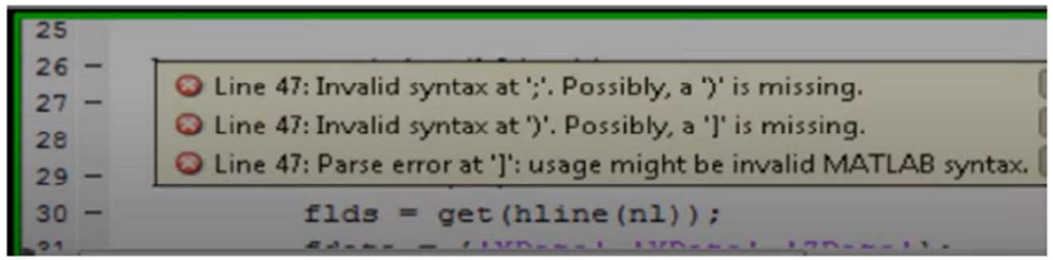
How do we debug the code? Steps of Debugging

- First step is to read the error
  - IDE will often print out an error message to the console
- Traverse to the line of code provided by the error
- use online forums such as StackOverFlow



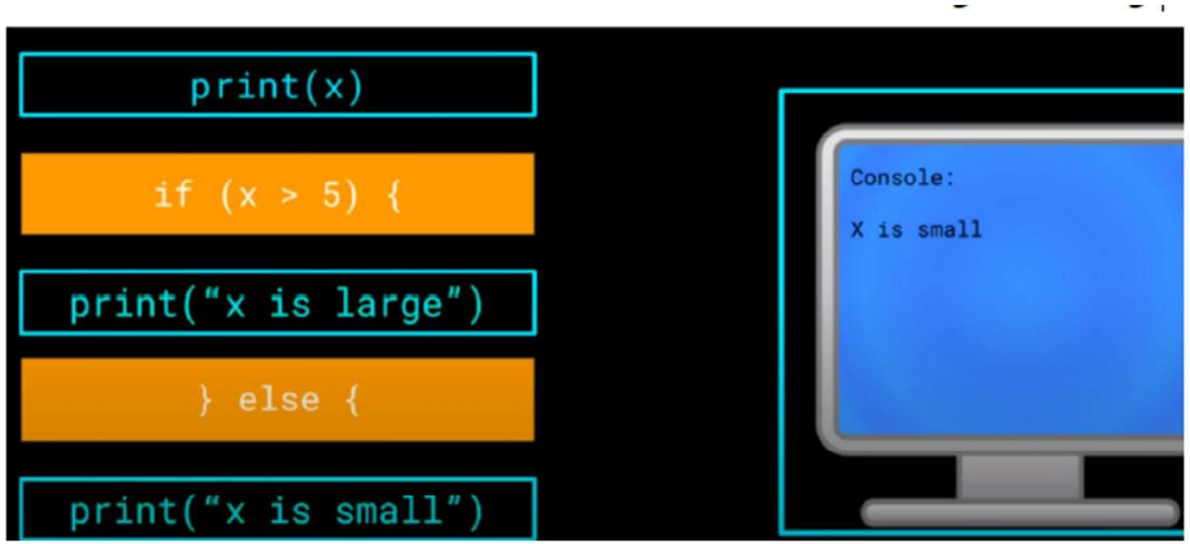
How do we debug the code? Steps of Debugging

- When a syntax or runtime error pops up, you should be able to find a fix for it fairly easily



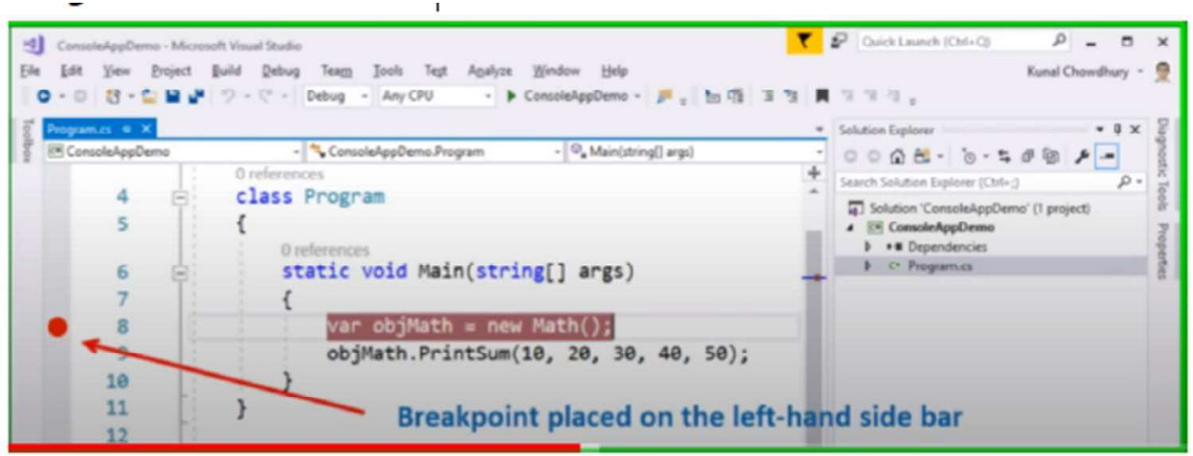
How do we debug the code? Print statements for debugging

- When a syntax or runtime error pops up, you should be able to find a fix for it fairly easily



How do we debug the code? Breakpoint for debugging

- Breakpoints
  - Pauses the program when the line you placed the breakpoint at is reached until you continue



How do we debug the code? fixing the bug

- When you think you've found the section of code causing the error
  - First try commenting out

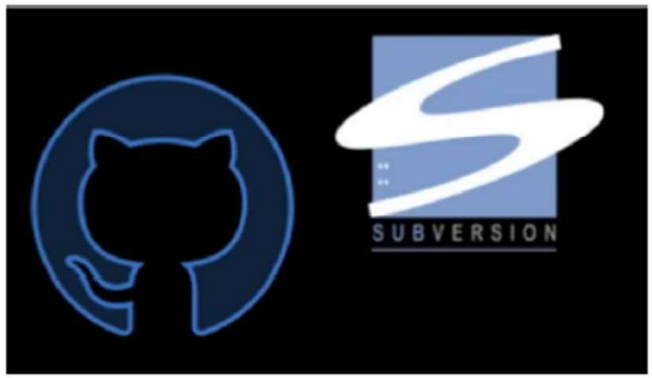
```
Line of code
Line of code
Commented Line of Code
Line of code
Line of code
Commented Line of Code
Commented Line of Code
Line of code
```

How do we debug the code? fixing the bug

- Commenting
  - Allows us to mark-up the code without the computer reading it as actual code
  - Essentially a documentation tool for programmers

How do we debug the code? Preventing the errors

- Backup the code frequently
- Version managers such as GitHub or SVN can help
- Useful for backtracking to find when the error was written



How do we debug the code? Preventing the errors

- Run your program frequently
  - Prevents you from saving a backup that doesn't work
  - Makes it easier to figure out when you wrote the error

Summary

In this lesson, you should have learned the following:

- The primary means of decision-making is the `if` statement, with the optional `else`.
- also offers the `switch` statement.
- Java provides three loop statements: `while`, `do…while`, and `for`.
- Use `break` and `continue` sparingly.
- Arrays Loops
- Errors and Debugging