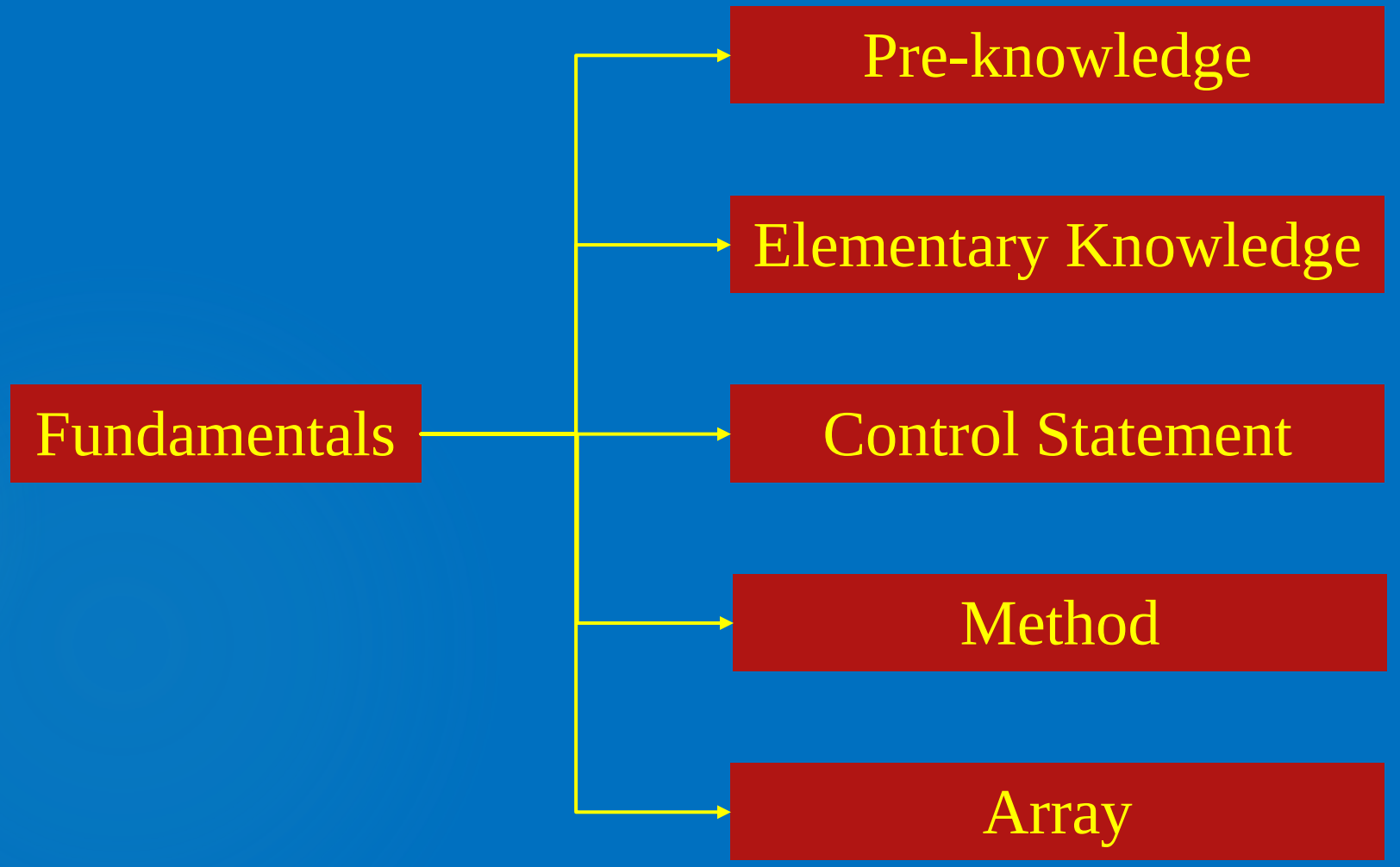
Review and Exam

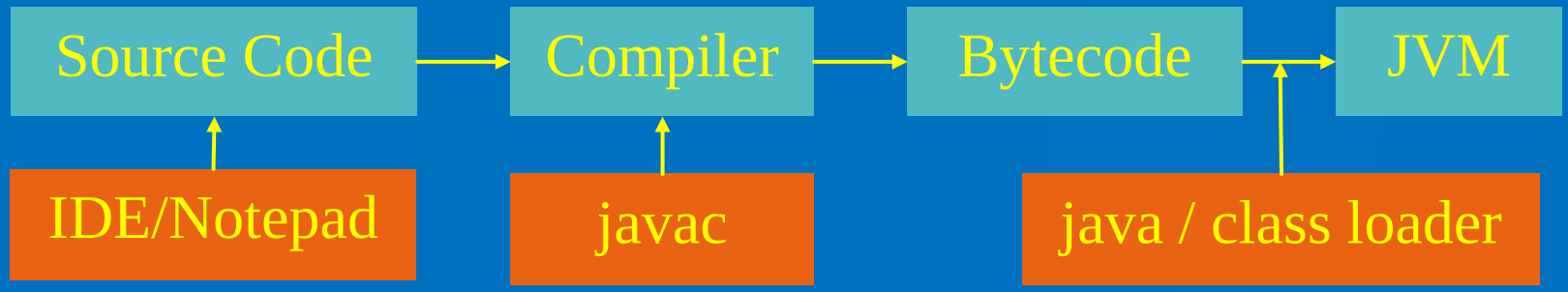

Fundamentals
Pre-knowledge
Elementary Knowledge
Control Statement
Method
Array

Pre-knowledge

Programming Languages

- Machine
- Assembly
- High-level

Programming Procedure



Program Structure

- Class
- Class Block
- Method
- Method Block

Elementary Knowledge

- **Identifiers**
  - Class Name
  - Method Name
  - Variable Name
  - Rules & Conventions
- **Variables**
  - Declare and initialize a variable
  - Scope of a variable
  - Assignment
  - Variable Types
    - byte, short, int, long, float, double, char, boolean
- **Constants**
- **Literals**
  - Default Type
  - 0B111,077,0XFF; 3.0F, 78L
- **Conversion**
  - Variable Types

Elementary Knowledge

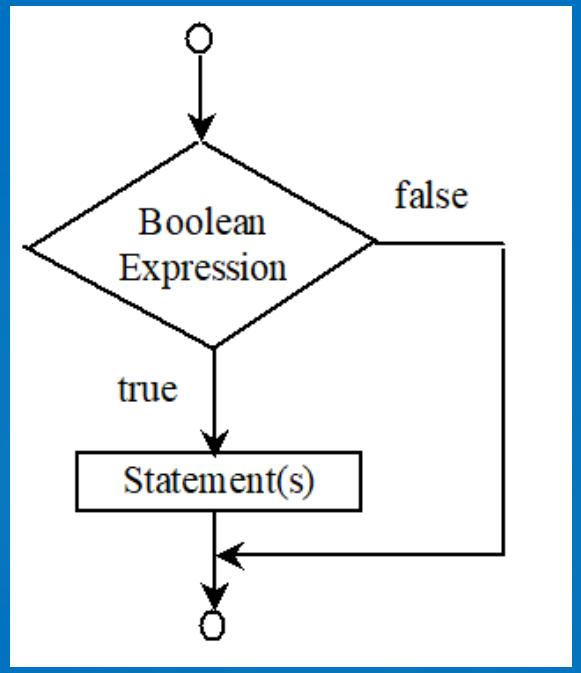
Arithmetic operation	+ - * / % ++ -- += -= *= /=
Comparison operation	> == < <= >= !=
Logical operation	|| && ! ^

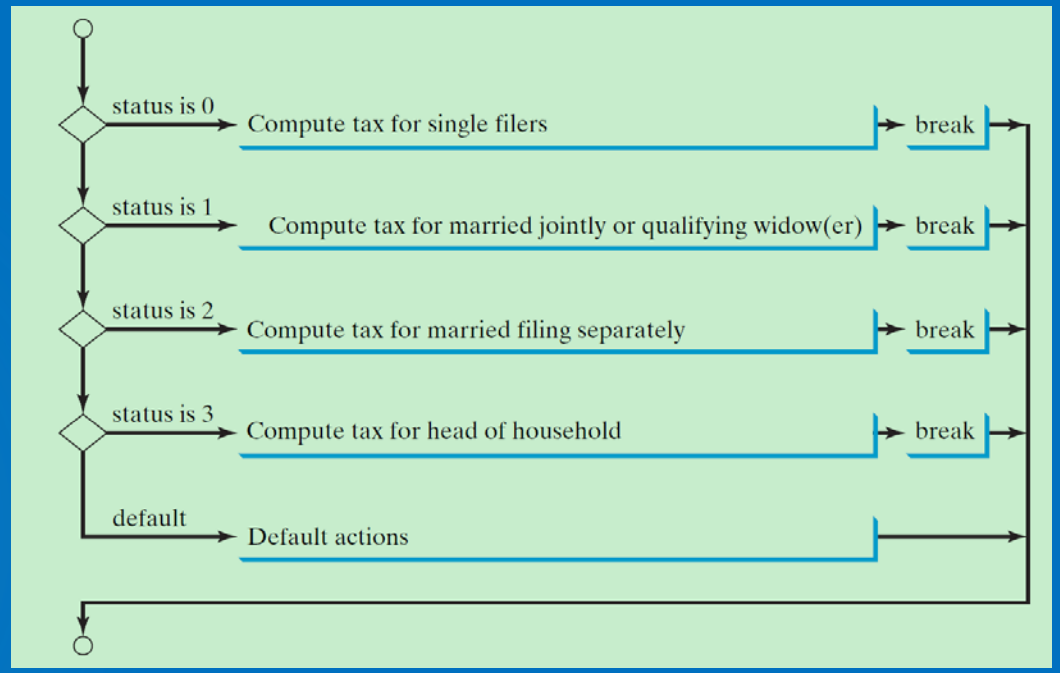
Control Statement

Selection

if-else

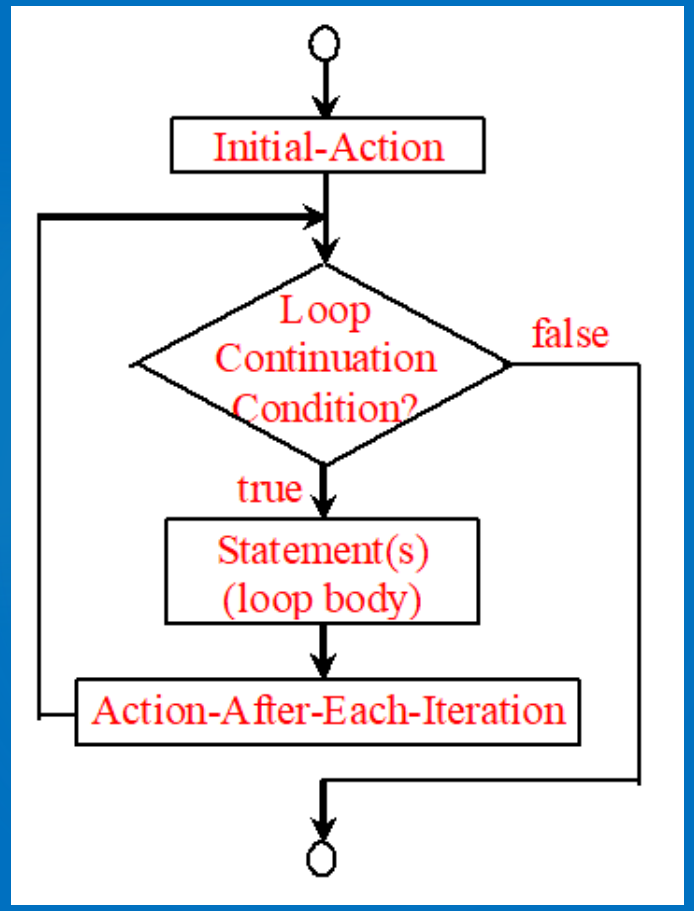
switch

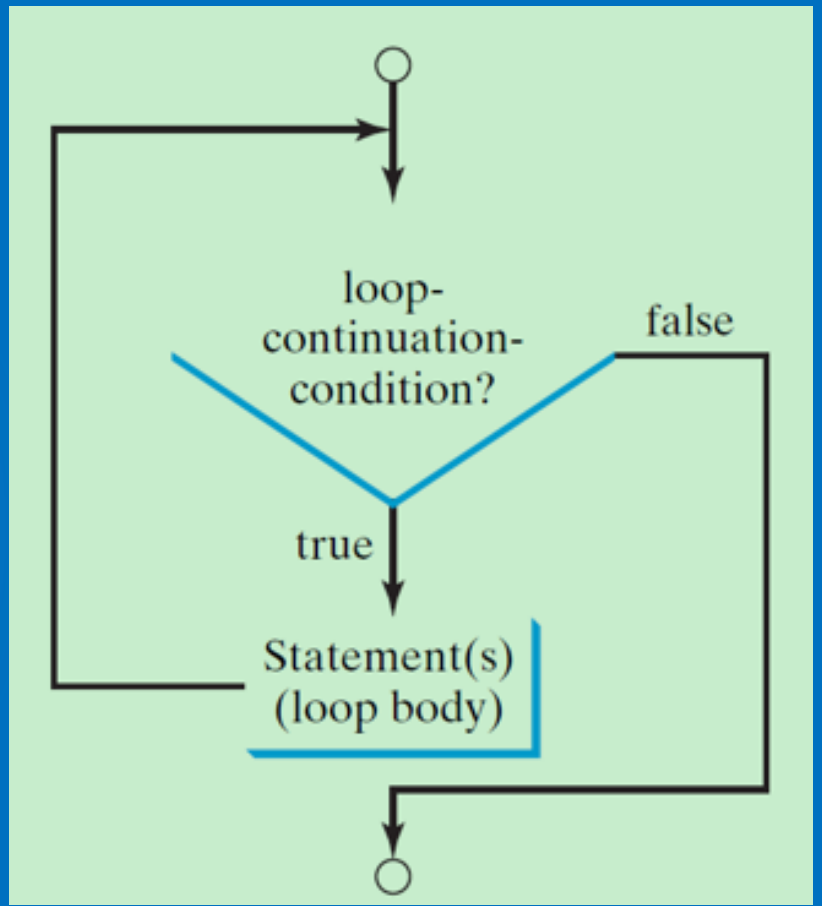


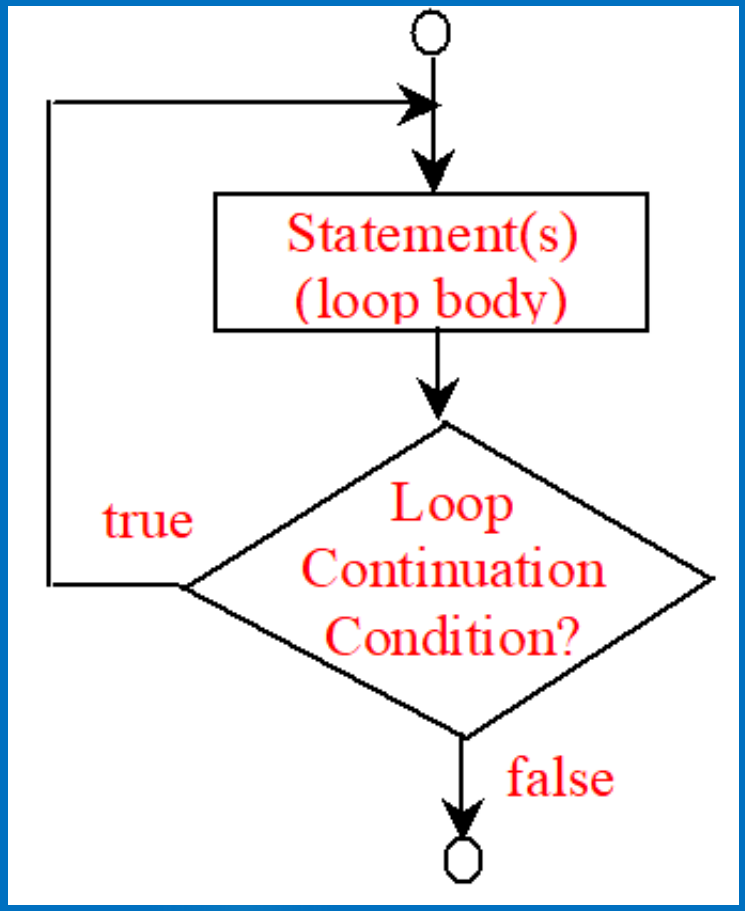


Control Statement

Loops | for | while/do-while







Method

Defining and Calling	Method Header	modifiers	Return value type
	Method Name	Formal Parameters	Actual Parameters
Passing Arguments	Passing Primary Type: int double		
	Passing Reference Type: Array / Object		
Local Variables	The scope of the local variables and initialization		

Array

Defining and Creating

```
int[] a = new int[10];
```

```
int[] a = {1,2,3,4};
```

```
int[] a = new int[]{1,2,3,4};
```

Arrays Copying

Using for statement

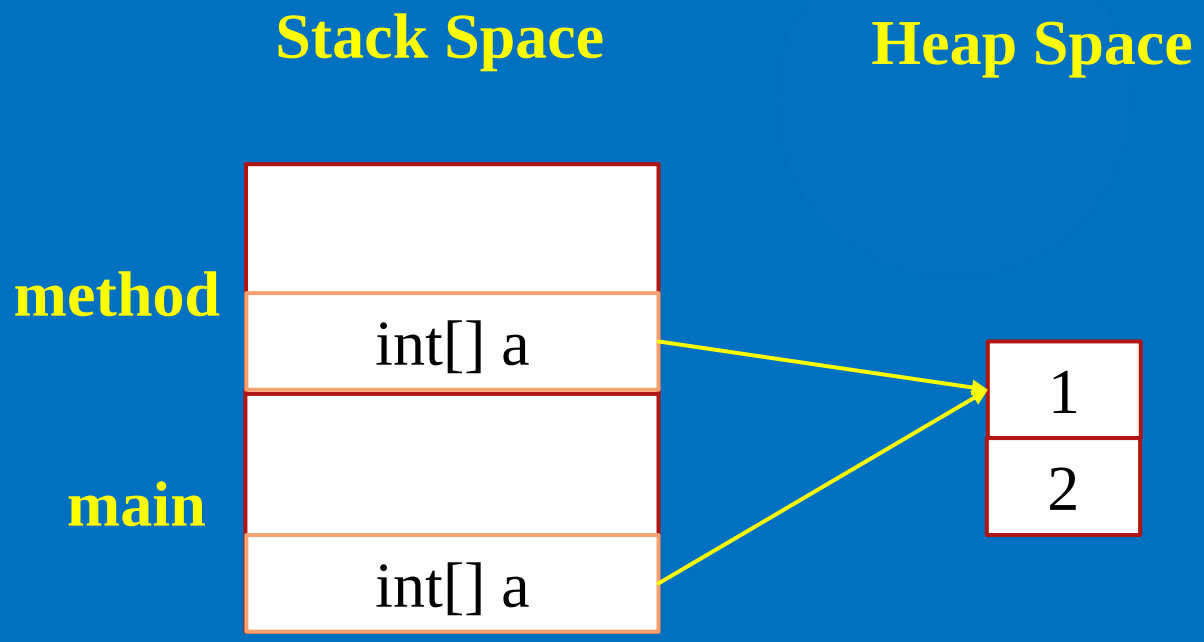
Using System.arraycopy()

Using Arrays.copyOf()

Array

Arrays Passing

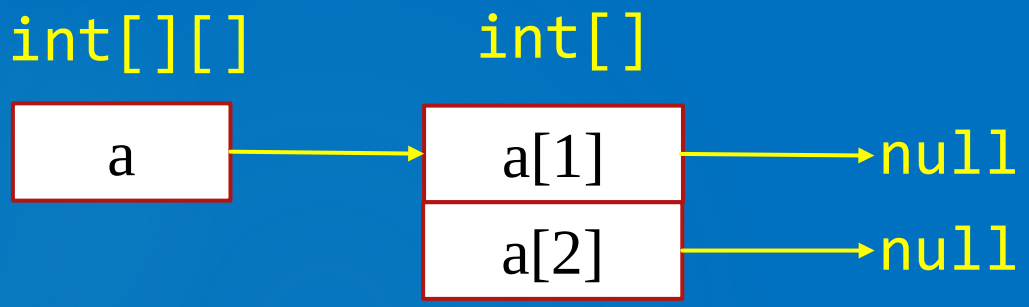
```
public … main(String[] args){
    int[] a = {1,2};
    method(a);
}
Public … method(int[] a){
}
```



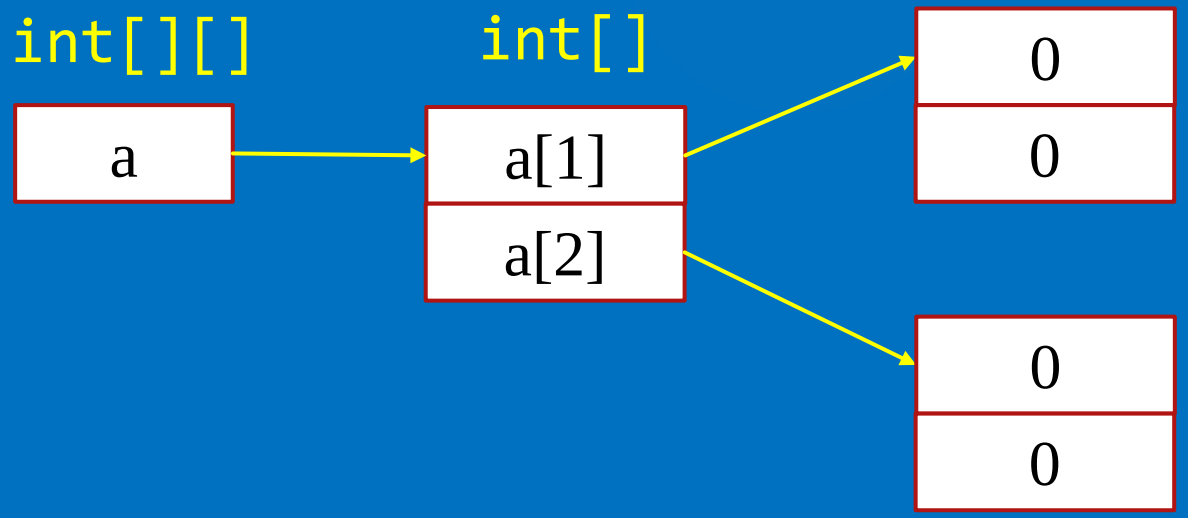
Array

Multidimensional Arrays

```
int[][] a = new int[2][];
```



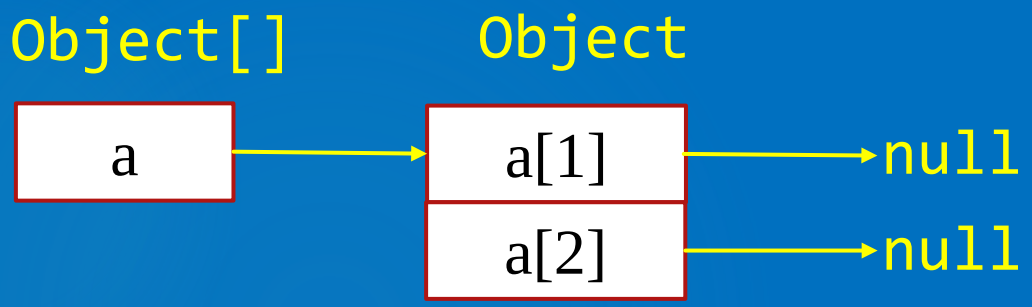

```
int[][] a = new int[2][2];
```

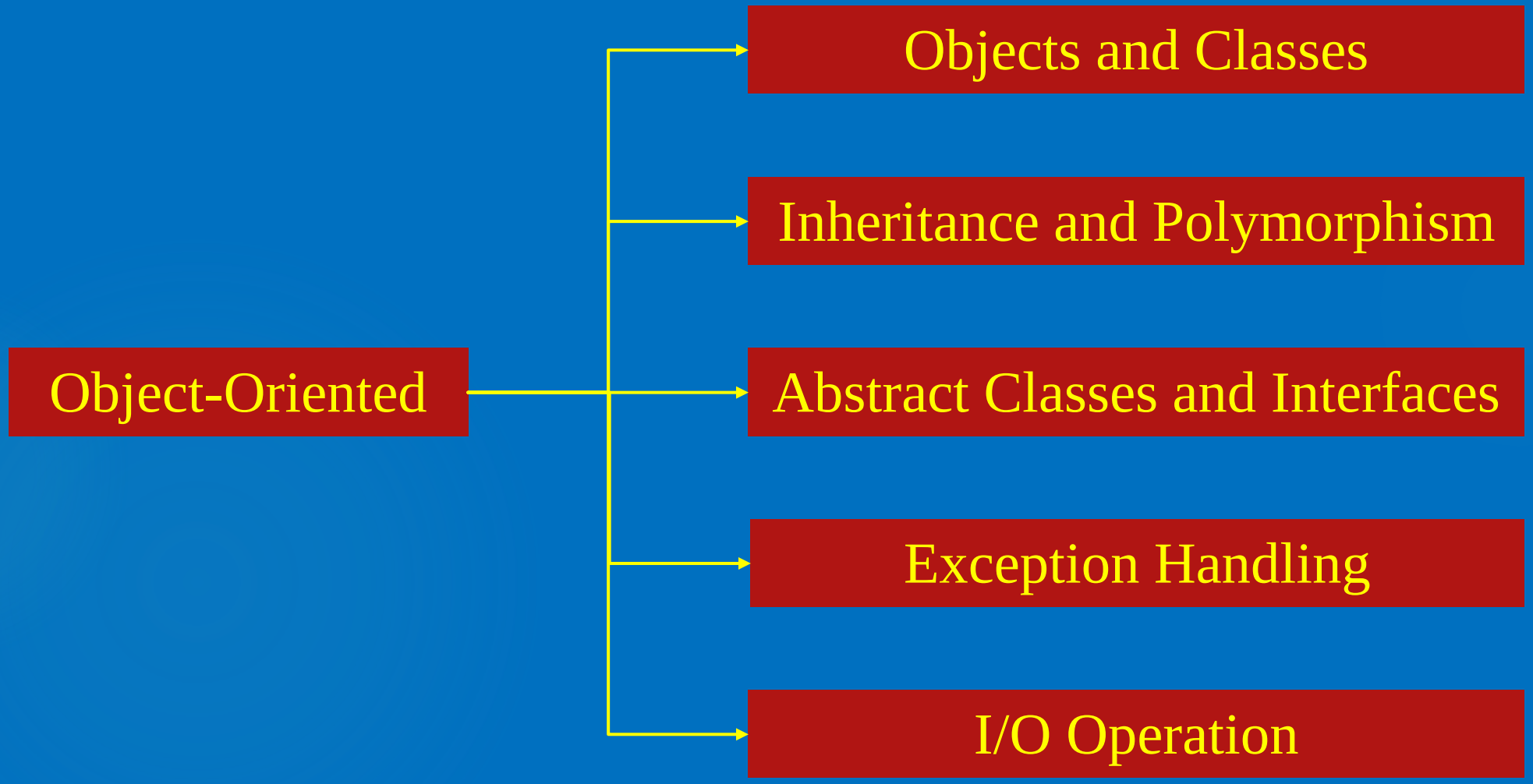


Array

Object Arrays

```
Object[] a = new Object[2];
```




Object-Oriented
Objects and Classes
Inheritance and Polymorphism
Abstract Classes and Interfaces
Exception Handling
I/O Operation

Objects and Classes

Defining Classes

```
class Circle {
    /** The radius of this circle */
    double radius = 1.0;

    /** Construct a circle object */
    Circle() {
    }

    /** Construct a circle object */
    Circle(double newRadius) {
      radius = newRadius;
    }

    /** Return the area of this circle */
    double getArea() {
      return radius * radius * 3.14159;
    }
}
```

Data fields

Constructors

Method

Objects and Classes

- Constructor
  - Default Constructor
  - this keyword
- Static Members
  - Static Variables
  - Static Methods
- Visibility
  - public
  - protected
  - package
  - private
- Passing Objects
  - Passing reference
- Encapsulation
  - private attributes
  - getter/setter

Objects and Classes

Wrapper Classes

Byte, Short, Integer, Long, Float, Double, Character, Boolean

Auto-boxing and Auto-unboxing

Immutable Classes

String

```
String s1 = “hi”; String s2 = new String(“hi”);
s1 == s2;
s1.equals(s2);
```

Inheritance and Polymorphism

Inheritance	Super-classes and Sub-classes	Super keyword
	Constructor Chain	Overriding: toString()
Polymorphism	Object type casting	Dynamic Binding
ArrayList		

Abstract Classes and Interfaces

Abstract Classes

```
public abstract class A{
   public abstract void method();
}
```

Interfaces

```
Public interface A{
   void method();
}
Public class B implements A{
   public void method(){}
}
```

Comparable

Cloneable

Exception Handling

Exception Types

Checked Exception

Unchecked Exception

Throw & Throws

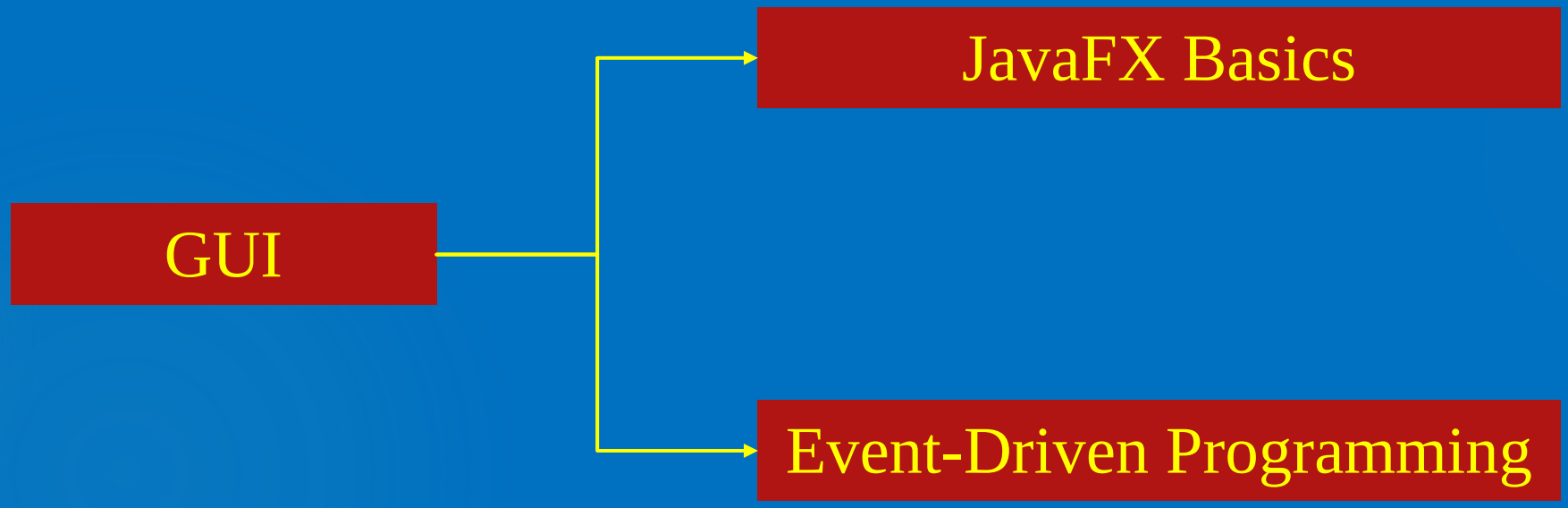
```
public void method()throws Exception{
   throw new Exception();
}
```

Try-Catch

```
Try{
   method();
}catch(Exception e){
   e.printStackTrace();
}finally{
   //
}
```

I/O Operation

Binary I/O	InputStream	OutputStream	FileInputStream	FileOutputStream
Text I/O	Scanner	PrintWriter		
	InputStreamReader	BufferredReader		
	OutputStreamWriter	BufferredWriter		
I/O Exception	FileNotFoundException	IOException		


GUI
JavaFX Basics
Event-Driven Programming

GUI

- Basics
  - Application
  - Stage
  - Scene
  - Node
  - Shape
  - ImageView
  - Control
  - Pane
- Event-Driven
  - Event Source
  - Event Object
  - Event Handler
  - EvenHandler<ActionEvent>

KEY TERMS

1. .class file
2. .java file
3. byte
4. bit
5. bytecode
6. block (class block/ method block)
7. comment
8. class loader
9. compiler
10. statement (statement terminator)
11. assembly language
12. high-level language
13. machine language

KEY TERMS

14. source code (source file)
15. Java Development Toolkit (JDK)
16. Java Virtual Machine (JVM)
17. operator (assignment operator)
18. debug (debugger)
19. declaration
20. backslash (\)
21. btye short int long float double char boolean
22. encode/decode
23. constant/variable/literal/identifier
24. casting
25. expression

KEY TERMS

26. syntax error/runtime error
27. selection statement (if…else/switch)
28. break statement
29. continue statement
30. do-while loop/ while loop
31. for loop
32. infinite loop
33. nested loop
34. formal parameters
35. actual parameters/ arguments
36. method overloading
37. return type/ value

KEY TERMS

38. pass-by-value
39. modifier
40. scope of variable
41. anonymous array
42. index
43. garbage collection
44. stack
45. this keyword
46. dynamic binding
47. constructor chaining
48. protected

KEY TERMS

49. inheritance
50. subclass
51. superclass
52. polymorphism
53. override
54. is-a relationship
55. abstract class
56. abstract method
57. multiple inheritance
58. interface
59. implements
60. deep copy/ shallow copy

KEY TERMS

62. wrapper class
63. AWT/ Swing/JavaFX
64. binding property
65. stage
66. scene
67. node
68. pane
69. shape
70. imageview
71. event
72. event source
73. event object

KEY TERMS

74. inner class
75. event handler
76. lambda expression

SUMMARY

1. A Java program must have a **main** method. The **main** method is the entry point where the program starts when it is executed.
2. Every statement in Java ends with a semicolon (;), known as the statement terminator.
3. Identifiers are names for things in a program. An identifier is a sequence of characters that consists of letters, digits, underscores (_), and dollar signs ($).
4. In Java, the equal sign (=) is used as the assignment operator.
5. A variable declared in a method must be assigned a value before it can be used.
6. A named constant is declared by using the keyword final.

SUMMARY

7. Integer arithmetic (/) yields an integer result.
8. The increment operator (++) and the decrement operator (—) increment or decrement a variable by 1.
9. Casting a variable of a type with a small range to a variable of a type with a larger range is known as widening a type.
10. Casting a variable of a type with a large range to a variable of a type with a smaller range is known as narrowing a type.
11. The character \ is called the escape character.
12. Programming errors can be categorized into three types: syntax errors, runtime errors, and logic errors.

SUMMARY

13. The Boolean operators &&, ||, !, and ^ operate with Boolean values and variables.
14. The switch statement makes control decisions based on a switch expression of type char, byte, short, or int. (String supported in Java 7)
15. The keyword **break** is optional in a switch statement, but it is normally 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 next case statement will be executed.

SUMMARY

16. A one-time execution of a loop body is referred to as an iteration of the loop.
17. The **break** keyword immediately ends the innermost loop, which contains the break.
18. The **continue** keyword only ends the current iteration.
19. The **method** header specifies the modifiers, return value type, method name, and parameters of the method.
20. When a program calls a method, **program control** is transferred to the called method. A called method **returns control** to the caller when its return statement is executed or when its method-ending closing brace is reached.

SUMMARY

21. Each time a method is invoked, the system stores **parameters** and **local variables** in a space known as a **stack**. When a method calls another method, the caller's stack space is kept intact, and new space is created to handle the new method call. When a method **finishes** its work and **returns** to its caller, its associated **space is released**.
22. A method can be **overloaded**. This means that two methods can have the **same name**, as long as their method **parameter** lists **differ**.

SUMMARY

23. A variable is declared as an array type using the syntax elementType[] arrayRef-Var or elementType arrayRefVar[]. The style elementType[] arrayRefVar is preferred, although elementType arrayRefVar[] is legal.
24. You cannot assign elements to an array unless it has already been created. You can create an array by using the new operator with the following syntax: **new element-Type[arraySize]**.
25. When an array is created, its elements are assigned the default value of 0 for the numeric primitive data types, '\u0000' for char types, and false for boolean types.

SUMMARY

26. Java has a shorthand notation, known as the array initializer, which combines in one statement declaring an array, creating an array, and initializing, using the syntax: **elementType[] arrayRefVar = {value0, value1, ..., valuek}**.
27. When you pass an array argument to a method, you are actually **passing the reference of the array**; that is, the called method can modify the elements in the caller's original array.
28. A variable for two-dimensional arrays can be declared using the syntax: elementType[][] arrayVar.

SUMMARY

29. An instance variable or method belongs to an instance of a class. Its use is associated with individual instances. A static variable is a variable shared by all instances of the same class. A static method is a method that can be invoked without using instances.
30. Modifiers specify how the class, method, and data are accessed. A public class, method, or data is accessible to all clients. A private method or data is accessible only inside the class.
31. You can provide a get method or a set method to enable clients to see or modify the data. Colloquially, a get method is referred to as a getter (or accessor), and a set method as a setter (or mutator).

SUMMARY

32. All parameters are passed to methods using pass-by-value. For a parameter of a primitive type, the actual value is passed; for a parameter of a reference type, the reference for the object is passed.
33. A Java array is an object that can contain primitive type values or object type values. When an array of objects is created, its elements are assigned the default value of **null**.
34. Strings are objects encapsulated in the String class. A string can be constructed using one of the constructors or using a string literal shorthand initializer.

SUMMARY

35. A String object is **immutable**; its contents cannot be changed. To improve efficiency and save memory, the JVM stores two literal strings that have the same character sequence in a unique object. This unique object is called an interned string object.
36. You can use the concat method to concatenate two strings, or the plus (+) sign to concatenate two or more strings.
37. You can use the equals and compareTo methods to compare strings. The equals method returns true if two strings are equal, and false if they are not equal. The compareTo method returns 0, a positive integer, or a negative integer, depending on whether one string is equal to, greater than, or less than the other string.

SUMMARY

38. You can pass strings to the main method from the command line. Strings passed to the main program are stored in args, which is an array of strings. The first string is represented by args[0], and args.length is the number of strings passed.
39. You can use Scanner to read string and primitive data values from a text file and use PrintWriter to create a file and write data to a text file.
40. A constructor is used to construct an instance of a class. Unlike properties and methods, the constructors of a superclass are not inherited in the subclass. They can be invoked only from the constructors of the subclasses, using the keyword **super**.

SUMMARY

41. Every class in Java is descended from the java.lang.Object class. If no inheritance is specified when a class is defined, its superclass is Object.
42. If a method's parameter type is a superclass (e.g., Object), you may pass an object to this method of any of the parameter's subclasses (e.g., Circle or String). When an object (e.g., a Circle object or a String object) is used in the method, the particular implementation of the method of the object that is invoked (e.g., toString) is determined dynamically.
43. It is always possible to cast an instance of a subclass to a variable of a superclass, because an instance of a subclass is always an instance of its superclass. When casting an instance of a superclass to a variable of its subclass, explicit casting must be used to confirm your intention to the compiler with the (SubclassName) cast notation.

SUMMARY

44. You can use obj instanceof AClass to test whether an object is an instance of a class.
45. The java.lang.Comparable interface defines the compareTo method. Many classes in the Java library implement Comparable.
46. The java.lang.Cloneable interface is a marker interface. An object of the class that implements the Cloneable interface is cloneable.
47. A class can extend only one superclass but can implement one or more interfaces.
48. An interface can extend one or more interfaces.
49. Exceptions occur during the execution of a method. RuntimeException and Error are unchecked exceptions; all other exceptions are checked.

SUMMARY

50. When declaring a method, you have to declare a checked exception if the method might throw it, thus telling the compiler what can go wrong. The keyword for declaring an exception is throws, and the keyword for throwing an exception is throw.
51. To invoke the method that declares checked exceptions, you must enclose the method call in a try statement. When an exception occurs during the execution of the method, the catch block catches and handles the exception.
52. The code in the finally block is executed under all circumstances, regardless of whether an exception occurs in the try block or is caught.