Dayananda Sagar College of Engineering

Department of Information Science and Engineering

Department of Information Science and Engineering

Course Name: INTRODUCTION TO JAVA

Course Code: 18IS6IEJVA
Semester: 6

Faculty: Mrs. Radhika T V
Assistant Professor, Dept of ISE
DSCE

Evolution of Java

- The development of each programming language is based on a fact: there is a need to solve a problem that was not resolved by previous programming languages.

- Programming languages such as Cobol, Fortran do not have structural principles.

- Therefore, C was invented in 1970, to replace the assembly language and to create a structured, effective and high-level language.

- C is a processor-oriented programming language.

- Though C was a quite efficient and successful programming language, the complexity of the program was seeking more efficient language to solve problems
- C++ came with object-oriented programming features.
- The Problem is how to control the software on different machines, a separate compiler is required for that CPU.

Differences between C, C++ and Java

Metrics	C	C++	Java
Programming Paradigm	Procedural language	Object-Oriented Programming (OOP)	Pure Object Oriented Oriented
Origin	Based on assembly language	Based on C language	Based on C and C++
Developer	Dennis Ritchie in 1972	Bjarne Stroustrup in 1979	James Gosling in 1991
Translator	Compiler only	Compiler only	Interpreted language (Compiler + interpreter)
Platform Dependency	Platform Dependent	Platform Dependent	Platform Independent
Code execution	Direct	Direct	Executed by JVM (Java Virtual Machine)
File generation	.exe files	.exe files	.class files

Differences between C, C++ and Java

Metrics	C	C++	Java
Pre-processor directives	Support header files (#include, #define)	Supported (#header, #define)	Use Packages (import)
keywords	Support 32 keywords	Supports 63 keywords	50 defined keywords
Datatypes (union, structure)	Supported	Supported	Not supported
Inheritance	No inheritance	Supported	Supported except Multiple inheritance
overloading	No overloading	Support Function overloading (Polymorphism)	Operator overloading is not supported
Pointers	Supported	Supported	Not supported
Allocation	Use malloc, calloc	Use new, delete	Garbage collector
Exception Handling	Not supported	Supported	Supported
Multithreading/ Interfaces	Not supported	Not supported	Supported

Sample C Programs

1. Program to add two integers

```
#include <stdio.h>
int main()
{
int number1, number2, sum;
printf("Enter two integers: ");
scanf("%d %d", &number1, &number2);
// calculating sum
sum = number1 + number2;
printf("%d + %d = %d", number1, number2, sum);
return 0;
}
```

Cntd..

2. **Program to Compute Quotient and Remainder**

```
#include <stdio.h>
 int main()
{
 int dividend, divisor, quotient, remainder;
printf("Enter dividend: ");
 scanf("%d", &dividend);
printf("Enter divisor: ");
scanf("%d", &divisor);
 // Computes quotient
quotient = dividend / divisor; // Computes remainder
 remainder = dividend % divisor;
 printf("Quotient = %d\n", quotient);
printf("Remainder = %d", remainder);
 return 0;
}
```

Object Oriented Programming using C++

- Object oriented programming is a type of programming which uses **objects** and **classes** for its functioning.
- **Objects** contain data in the form of attributes and code in the form of methods.
- Attributes and methods are basically variables and functions that belongs to the class-> Class Members.
- A **class** is a user-defined data type that we can use in our program, and it works as an object constructor, or a "blueprint" for creating objects.
- It is based on real world entities like inheritance, polymorphism, data hiding, etc

Key Concepts of Object Oriented Programming

- There are a few principle concepts that form the foundation of object-oriented programming

  - **Object**
  - **Class**
  - **Abstraction**
  - **Encapsulation**
  - **Inheritance**
  - **Polymorphism**
  - **Overloading**

Creation of Class and object

1. Create a class

```
class MyClass  // The class
{
    public:            // Access specifier
      int myNum;       // Attribute (int variable)
      string myString;  // Attribute (string variable)
};
```

2. Create a Object

- To create an object of MyClass, specify the class name, followed by the object name. MyClass.a
- To access the class attributes (myNum and myString), use the dot syntax (.) on the object:

Example-1

Create an object called "myObj" and access the attributes:

```
#include <iostream>
class MyClass {       // The class
  public:             // Access specifier
    int myNum;        // Attribute (int variable)
    string myString;  // Attribute (string variable)
};

int main() {
  MyClass myObj;  // Create an object of MyClass

  // Access attributes and set values
  myObj.myNum = 15;
  myObj.myString = "Some text";

  // Print attribute values
  cout << myObj.myNum << "\n";
  cout << myObj.myString;
  return 0;
}
```

Output:

15

Some text

Example-2

Creation of Multiple Objects:

```
#include <iostream>
#include <string>
using namespace std;

class Car {
  public:
    string brand;
    string model;
    int year;
};

int main() {
  Car carObj1;
  carObj1.brand = "BMW";
  carObj1.model = "X5";
  carObj1.year = 1999;
```

Cntd..

```
Car carObj2;
  carObj2.brand = "Ford";
  carObj2.model = “Fiesta";
  carObj2.year = 2007;


  cout << carObj1.brand << " " << carObj1.model << " " << carObj1.year
     << "\n";
  cout << carObj2.brand << " " << carObj2.model << " " << carObj2.year
     << "\n";
  return 0;
}
```

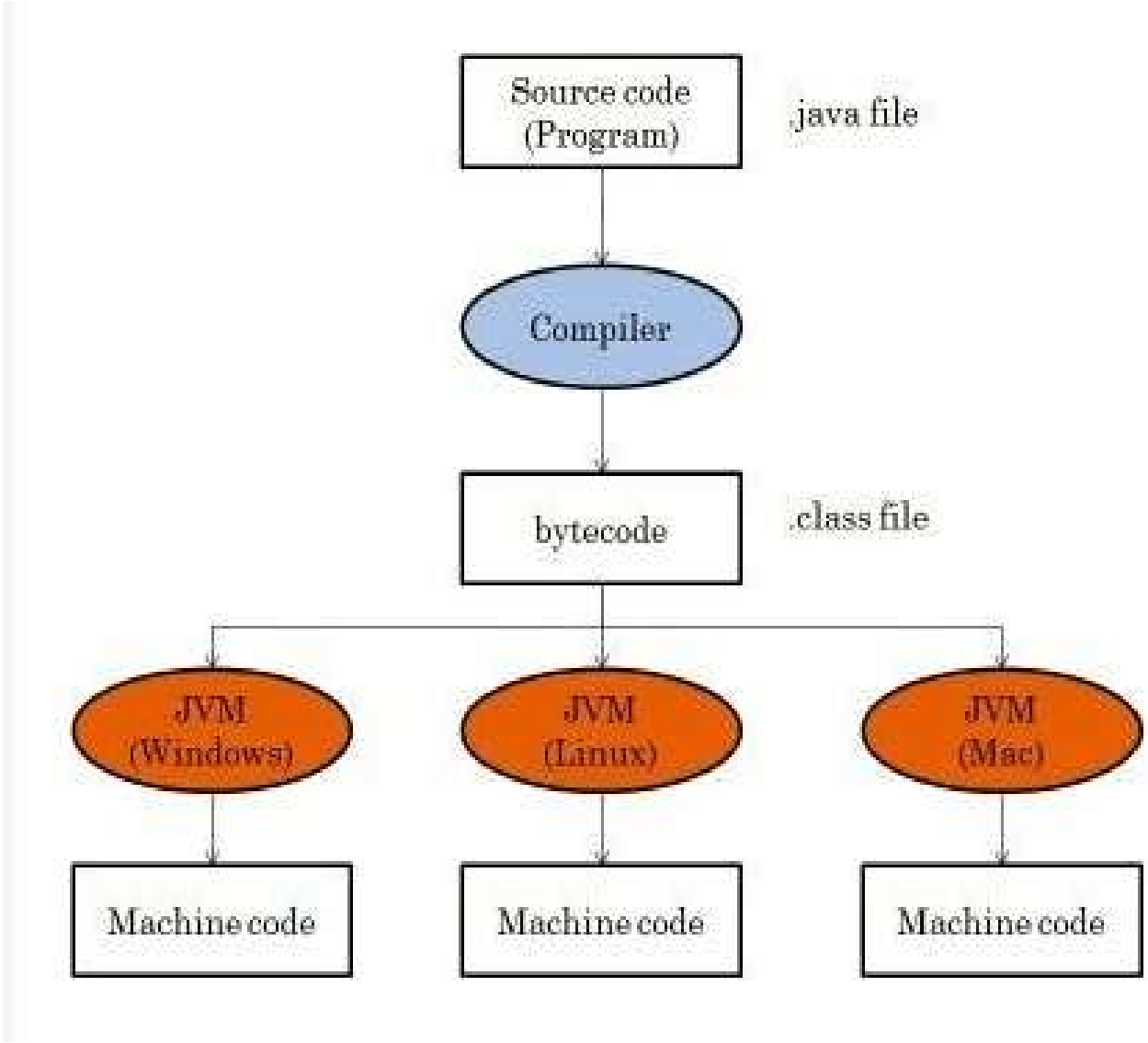
Output:

BMW X5 1999
Ford Fiesta 2007

Introduction:

- Java is a high-level programming language developed by Sun Microsystems and released in 1995.
- James Gosling initially developed Java in **Sun Microsystems** (which was later merged with **Oracle Corporation**).
- Java is a set of features of C and C++.
- Java programs are platform independent.
- As soon as a java program is compiled, java byte code is generated. Resources required to run the byte code are made available by the Java Virtual Machine(JVM).
- Java code that runs on one platform does not need to be recompiled to run on another platform; it's called **write once, run anywhere(WORA)**.

Java Virtual Machine



Key Features of Java

- Object Oriented
- Platform Independent
- Simple
- Secure
- Architecture-neutral
- Portable
- Robust
- Multi-threaded
- Distributed

Example

```
  /* FileName : "HelloWorld.java". */
class HelloWorld
{
    // Your program begins with a call to main().
    // Prints "Hello, World" to the terminal window.
    public static void main(String args[])
    {
        System.out.println("Hello, World");
    }
}
```

Course Outcomes- INTRODUCTION TO JAVA [18IS6IEJVA]

CO1 - Articulate classes, its members and the relationships among them needed for a specific problem.

CO2- Apply the basic concepts of object oriented programming in writing java programs.

CO3-Create and use packages/ interfaces in Java programs.

CO4-Analyze and implement exception handling in Java.

CO5-Use various Input/output packages effectively.

CO6-Design programs using String libraries.

SAMPLE PROGRAMS

Program 1

```
/*
Here is short example.
*/
class Example1 {
public static void main(String args[]) {
int num;
num = 100;
System.out.println("This is num: " + num);
num = num * 2;
System.out.print("The value of num * 2 is ");
System.out.println(num);
}
}
```

Output:

When you run this program, you will see the following output:

This is num: 100

The value of num * 2 is 200

Program 2

Program 2:
Write a Program that illustrates the if statement:

```
/*
Demonstrate the if.
Call this file "IfSample.java".
*/
class IfSample {
public static void main(String args[]) {
int x, y;
x = 10;
y = 20;
if(x < y) System.out.println("x is less than y");
x = x * 2;
if(x == y) System.out.println("x now equal to y");
x = x * 2;
if(x > y) System.out.println("x now greater than y");
// this won't display anything
if(x == y) System.out.println("you won't see this");
}
}
```

Output:

```
x is less than y
x now equal to y
x now greater than y
```

Program 3

Program 3:
Write a Program that illustrates the for loop:

```
/*
Demonstrate the for loop.
Call this file "ForTest.java".
*/
class ForTest {
public static void main(String args[]) {
int x;
for(x = 0; x<10; x = x+1)
System.out.println("This is x: " + x);
}
}
```

Output:

```
This is x: 0
This is x: 1
This is x: 2
This is x: 3
This is x: 4
This is x: 5
This is x: 6
This is x: 7
This is x: 8
This is x: 9
```

Program 4

Write a Program to illustrate one dimensional Array

```
class Array {
public static void main(String args[]) {
int month_days[];
month_days = new int[12];
// int month_days[] = new int[12];
month_days[0] = 31;
month_days[1] = 28;
month_days[2] = 31;
month_days[3] = 30;
month_days[4] = 31;
month_days[5] = 30;
month_days[6] = 31;
month_days[7] = 31;
month_days[8] = 30;
month_days[9] = 31;
month_days[10] = 30;
month_days[11] = 31;
System.out.println("April has " + month_days[3] + " days.");
```

Output:

April has 30 Days

Program 5

Write a Program to demonstrate the basic arithmetic operators.

```
class BasicMath {
public static void main(String args[]) {
// arithmetic using integers
System.out.println("Integer Arithmetic");
int a = 1 + 1;
int b = a * 3;
int c = b / 4;
int d = c - a;
int e = -d;
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
System.out.println("d = " + d);
System.out.println("e = " + e);
// arithmetic using doubles
System.out.println("\nFloating Point Arithmetic");
double da = 1 + 1;
double db = da * 3;
double dc = db / 4;
```

Output:

```
Integer Arithmetic
a = 2
b = 6
c = 1
d = -1
e = 1
Floating Point Arithmetic
da = 2.0
db = 6.0
dc = 1.5
```