CYBAGE
Delivering Value. Scientifically.

Introduction to Java

Author & Presenter -Asfiya Khan
(Training Specialist-Technical)

www.cybage.com

Agenda

- History of Java
- Features of Java
- Data types in Java
- Access Modifiers
- Writing first Java Class
- Accessors and Mutators
- Constructors
- ‘this’ keyword

History of Java

- Background : Sun Microsystems set up a project called “Project Green” to develop a platform independent language for embedded systems.
- The Language was first named as “OAK”.
- Then it was renamed as “Java” (One programming language with this name was already in existence)
- Java was dismissed earlier but again gained popularity when WWW became popular
- Though it is associated with the World Wide Web, it is older than the origin of Web.

Features of Java

- Simple
- Object Oriented
- Architecture Neutral
- Portable
- Robust
- Interpreted
- Distributed
- Dynamic
- Secure
- Multithreaded

Simple

- No Header files
- No Pointer arithmetic
- No Operator overloading
- Syntax similar to C++

Robust

- Memory management is done by the system.
- Developer need not have to worry about problems associated with pointers like:

  - Bad Pointers

  - Memory Leakage

- Strong Exception Handling mechanism that includes Compile time and dynamic checking.

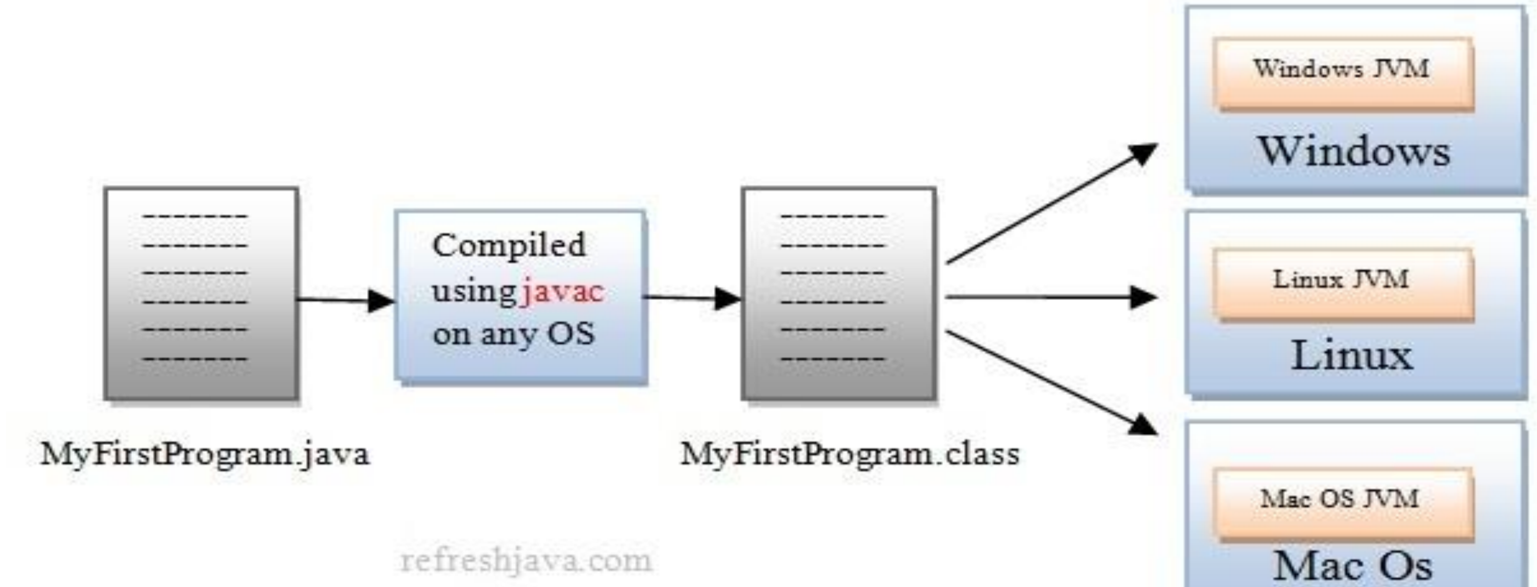
Architecture Neutral

Output of compilation of a .java file /java source code is a .class file.

• It is also called as Bytecode.

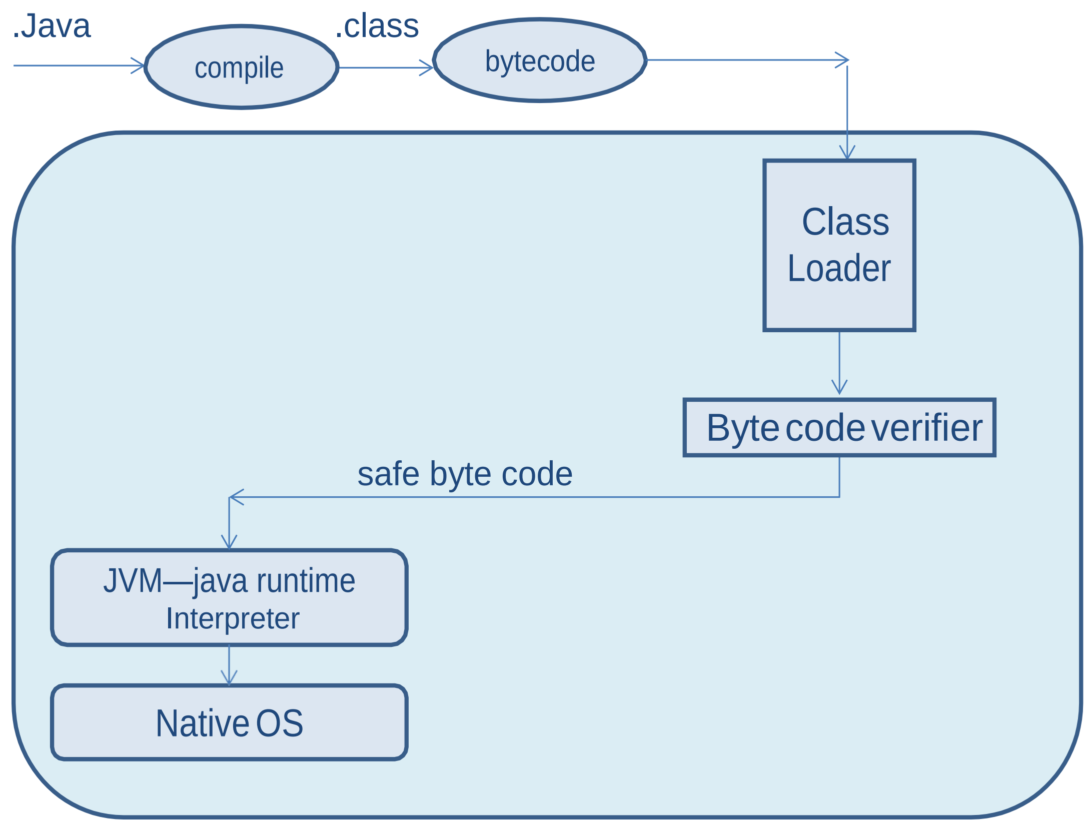
• Generated bytecode is platform independent which can be transferred to any particular platform / os.

Java : Platform Independent



CYBAGE

Java Environment



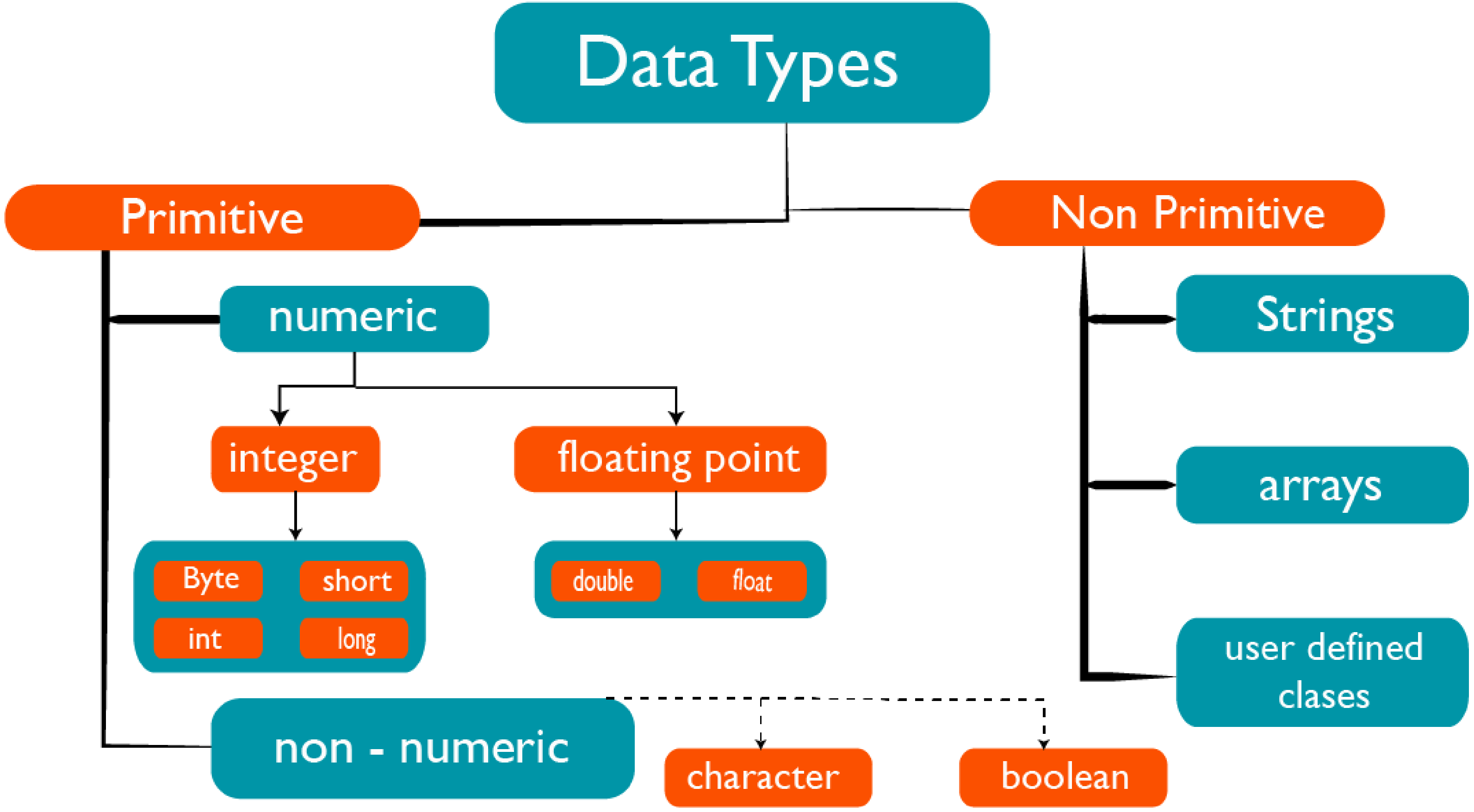
Secured, Interpreted

Secure -

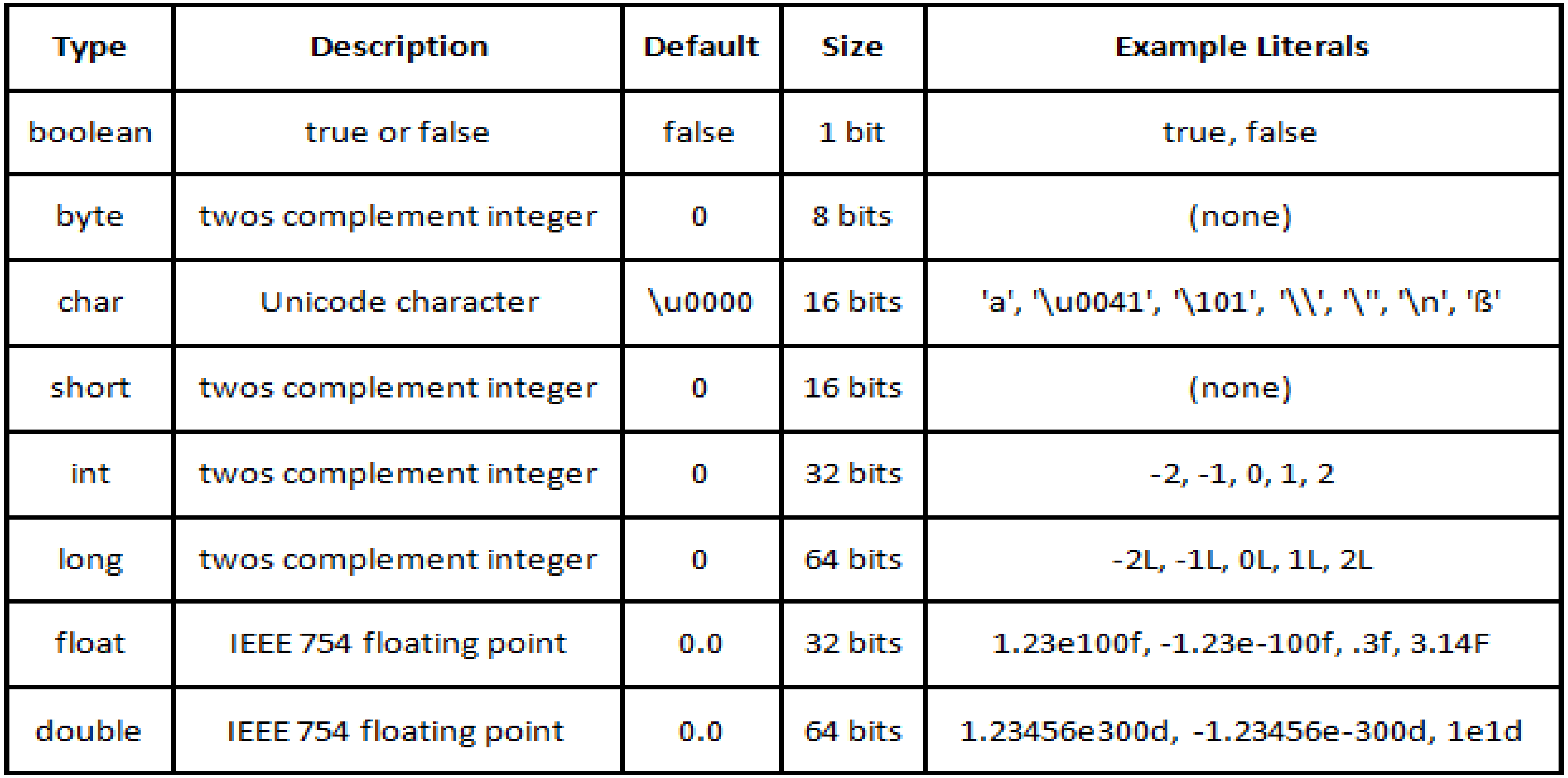
- Java is intended for use in networked/distributed environments.
- Therefore a lot of emphasis has been placed on security.
- Java enables the construction of virus-free, tamper-free systems.
- Interpreted Java byte codes are translated on the fly to native machine instructions (interpreted).
- classes are linked on need basis.

Portable

- The sizes of the primitive data types are specified.
- Behavior of basic datatype sizes & arithmetic operators is consistent across the platforms.
- For example, "int‟ always a 32 bit integer.
- Standard Unicode format is used for storing Strings.


Data Types
Primitive
Non Primitive
numeric
integer
floating point
Byte
short
int
long
double
float
non - numeric
character
boolean
Strings
arrays
user defined clases

Default Values of Datatype



Type	Description	Default	Size	Example Literals
boolean	true or false	false	1 bit	true, false
byte	twos complement integer	0	8 bits	(none)
char	Unicode character	\u0000	16 bits	'a', '\u0041', '\101', '\\', '\'', '\n', 'ß'
short	twos complement integer	0	16 bits	(none)
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d

Access Modifiers

As the name suggests access modifiers in Java helps to restrict the scope of a class, constructor , variable , method or data member. There are four types of access modifiers available in java:

- default – No keyword required
- private
- protected
- public

Access Modifiers

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Writing First Java Class

```
public class Employee {
                int age;
                String name, desgn;
public void setEmployeeDetails(int age, String name,
String desgn)  {
                this.age = age;
                this.name = name;
                this.desgn = desgn;
}
public void getEmployeeDetails() {
System.out.println("Employee Data is=> Name:
"+this.name+" and Designation is: "+this.desgn);
                }
}
```

```
public class Test {

public static void main(String[] args) {
  Employee firstEmp=new Employee();
 firstEmp.setEmployeeDetails(34,"John","SE");
 firstEmp.getEmployeeDetails();


                }

}
```

Class is a blueprint for Object

In software, a class is a description of an object:

- A class describes the data that each object includes.
- A class describes the behaviors that all objects exhibits.
- A class represents the structure of the object
- An object is called as an instance of class

Accessors and Mutators

- Data is encapsulated inside an object
- Methods are required to set, access or to modify this data.
- Mutators or Setters – the methods to set data into an object.
- Naming convention:

  public void setXXX(…){}
- Accessors or Getters :The methods to access the data from an object
- Naming convention :

  public datatype getXXX(){}

Accessing Object Members

Accessing Object Members

The dot notation is: <object>.<member>

This is used to access object members, including attributes and methods.

Examples of dot notation are:

```
d.display();
d.age = 42;
```

Accessors and Mutators

```
class Employee
{
     int age;
     Stirng name;

 public void setAge(int a)  {   // setter or mutator
           this.age=a;
}

 public int getAge() //getter or accessor {
         return age;
  }
}
```

Constructor

Constructor is a special method:

- Its name is same as class name
- Constructor does not have any return type
 (not even void)
- It gets invoked implicitly whenever a new object is created
- Constructors can be overloaded

Default Constructor

The Default Constructor

- There is always at least one constructor for every class
- If the programmer does not supply any constructor explicitly, the default constructor will be created and executed implicitly.
- The default constructor takes no parameters
- The default constructor body is empty.

Constructor With Parameter

You can pass parameters to a constructor.

Example:

```
public class Employee
{
        private int age;

        public Employee(int age)
        {
                age = 42;
        }
}
```

“this” Keyword

- “this” is a keyword in java
- it points to the current invoking object
- every class member gets a hidden reference – ”this”
- For d1.display() or d1.dd :

  here current invoking object is “d1” so ‘this’ points to d1

Demo : ‘this’

```
class Machine
{
        String modelName;

         public Machine()
        {
                System.out.println(“To automate users tasks. ”);
        }
         public Machine(String name)
        {
                this(); //……constructor chaining
                this.name=name;
        }
}
```

ANY
QUESTIONS
?

CYBAGE
Delivering Value. Scientifically.

Thank You!

www.cybage.com