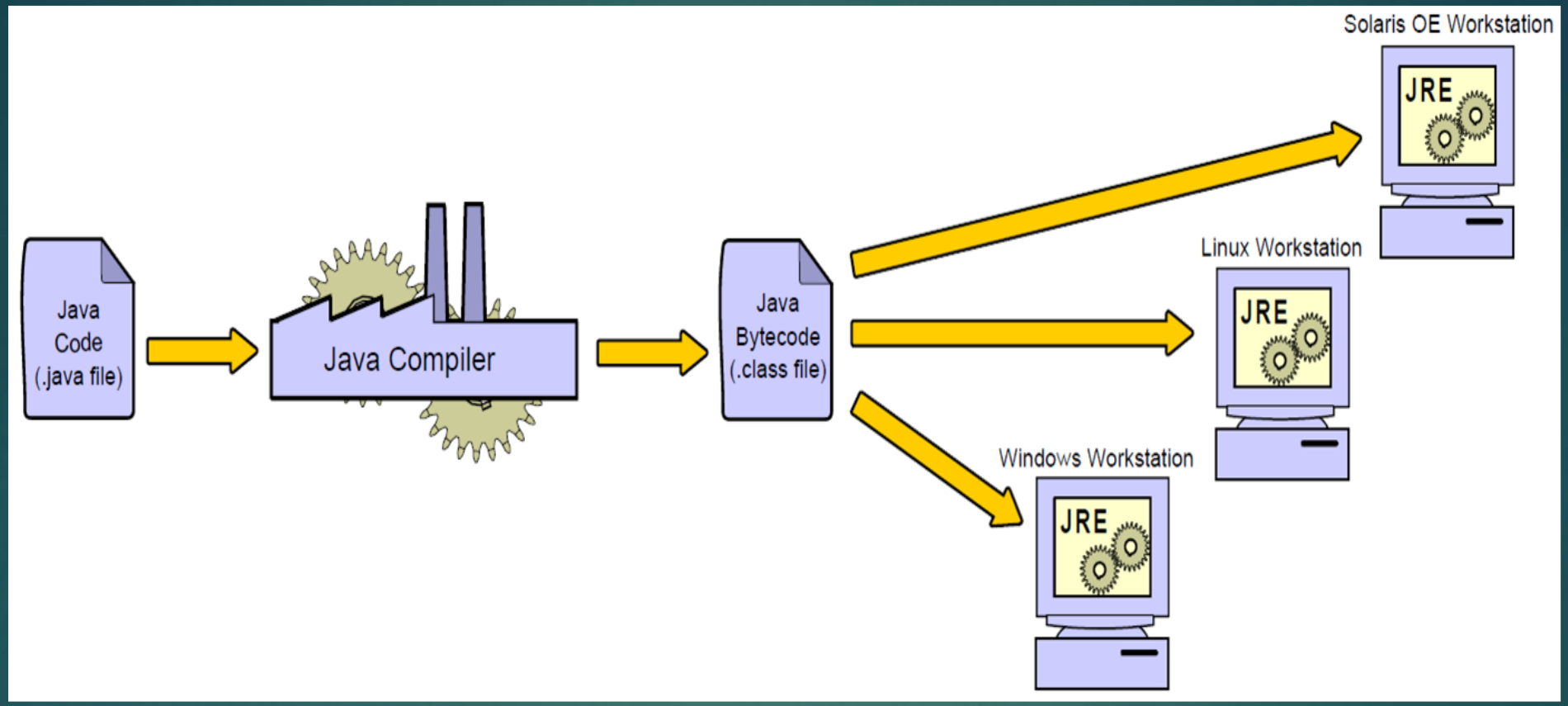
Objectives

- **In this session, you will learn to:**
  - Explore the Java platforms and versions
  - Explore the open nature of Java and its community
  - Create simple Java classes
  - Create primitive variables
  - Use Java SE 7 numeric and binary literals
  - Use if-else branching statement

Java Programs Are Platform-Independent

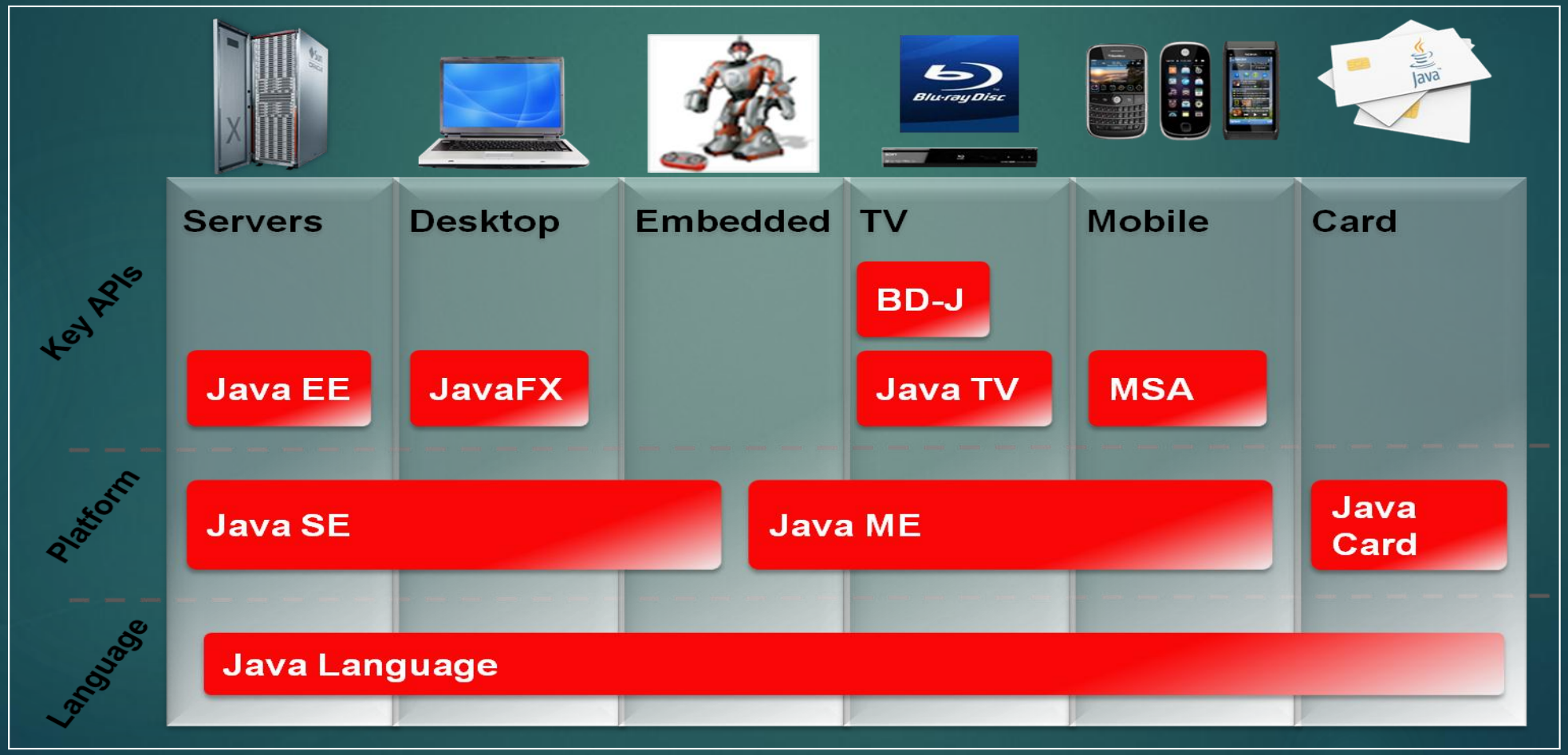
- Java technology applications:
  - Compiled to Java bytecode using Java SE Development Kit (JDK)
  - Bytecode gets executed on the Java platform
- Java Runtime Environment (JRE):
  - Runnable Java platform
  - Makes Java platform-independent

Java Programs Are Platform-Independent (Contd.)



Java Technology Product Groups

- The following diagram depicts different product groups used by the Java technology.



Java SE Platform Versions

Year	Developer Version (JDK)	Platform
1996	1.0	1
1997	1.1	1
1998	1.2	2
2000	1.3	2
2002	1.4	2
2004	1.5	5
2006	1.6	6
2011	1.7	7

Java Language Review

- To work on Java, you need to be familiar with the following concepts:
  - The basic structure of a Java class
  - Program block and comments
  - Variables
  - Basic `if-else` and `switch` branching constructs
  - Iteration with `for` and `while` loops

Class Structure

- The following code snippet shows the syntax of a basic class in a Java program:

```
package <package_name>;
   import <other_packages>;
public class ClassName
{
<variables(also known as fields)>;
<constructor method(s)>;
<other methods>;
}
```

A Simple Class

- The following code snippet shows a simple Java class with the `main()` method:

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

}
}
```

Code Blocks

- In Java:
  - The class and method declarations are enclosed in a code block.
  - Fields and methods have a block (or class) scope.
  - Code blocks are defined in braces `{ }`, as shown in the following example:

```
public class SayHello {
    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

Primitive Data Types

- The following table lists the primitive data types used in Java.

Category	Integer	Floating Point	Character	True False
Data Types	byte short int long	float double	char	boolean
Examples	1, 2, 3, 42 07 0xff	3.0F .3337F 4.022E23	'a' '\u0061' '\n'	true false
Default Values	0	0.0	'\u0000'	false

Java SE 7 Numeric Literals

- To improve readability, numeric literals can have any number of underscore characters (_) between digits.
- The following table shows the comparison of numeric literals in the earlier versions and the current version of Java.

Prior to Java 7	*In Java 7*
`long cardNo = 1234567890123456L;`	`long cardNo = 1234_5678_9012_3456L;`
`long socialSecurityNumber = 999999999L;`	`long socialSecurityNumber = 999_99_9999L;`
`long hexWords = 0xCAFEBABE;`	`long hexWords = 0xCAFE_BABE;`
`long maxLong = 0x7fffffffffffffffL;`	`long maxLong = 0x7fff_ffff_ffff_ffffL;`

Java SE 7 Binary Literals

- Binary literals:
  - Are Java integer values.
  - Can be expressed using the binary system by adding the prefixes `0b` or `0B` to the number.
- The following table shows the comparison of binary literals in the earlier versions and the current version of Java.

Earlier Versions	In Java 7
`byte aByte = 00100001;`	`byte aByte = 0b0010_0001;`
`short aShort = (short)101000010100;`	`short aShort = (short)0b1010_0001_0100;`

`int anInt2 = 101;`	`int anInt2 = 0b101;`
	
	`or`
	
	`int anInt3 = 0B101;`
	`//B is upper or lower case`