Object Oriented Programming

Dr. D. Sudheer,
Asst. Prof. Sr.
SCOPE
VIT-AP

Objectives

- **To design the concepts of object-oriented, event driven, and concurrent programming paradigms and develop skills by using these paradigms in Java.**
- **To analyze, design the principals of inheritance, dynamic polymorphism and interfaces.**
- **To learn writing a computer program to solve specified problems.**
- **To enable using the Java SDK environment to create, debug and run simple applications.**

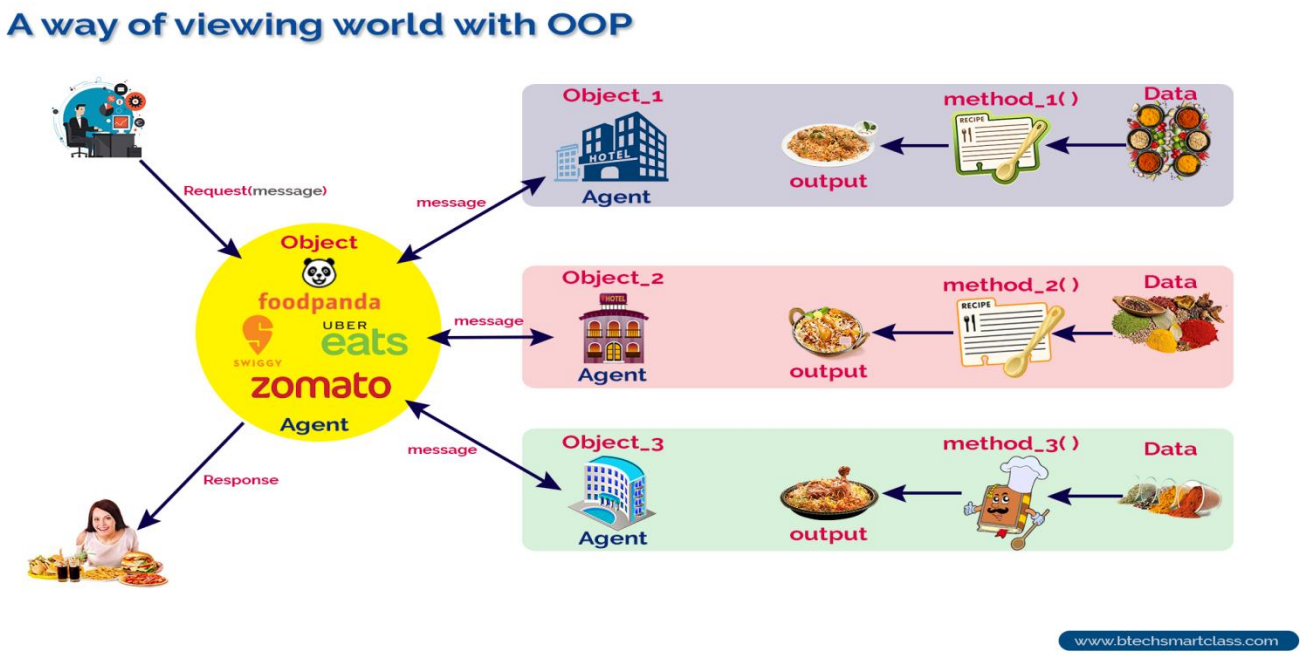
Expected Outcome

- **Design the structure of the Java programming language**
- **Identify classes, objects, members of a class and relationships among them needed for a specific problem**
- **Develop applications using packages, interfaces and also database connection.**
- **Develop Java programs to implement error handling techniques using exception handling**
- **Develop applications using Object Oriented Programming principals and proper programming structure**
- **Develop and understand multithreaded applications with synchronization**

Object-Oriented Thinking

A way of viewing world:

- A way of viewing the world is an idea to illustrate the object-oriented programming concept with an example of a real-world situation.



Agent and Communities

- To solve my food delivery problem, I used a solution by finding an appropriate agent (Zomato) and pass a message containing my request. It is the responsibility of the agent (Zomato) to satisfy my request. Here, the agent uses some method to do this. I do not need to know the method that the agent has used to solve my request. This is usually hidden from me.
- **An object-oriented program is structured as a community of interacting agents, called objects. Where each object provides a service (data and methods) that is used by other members of the community.**

Messages and Methods

- To solve my problem, I started with a request to the agent zomato.
- **In object-oriented programming, every action is initiated by passing a message to an agent (object), which is responsible for the action. The receiver is the object to whom the message was sent. In response to the message, the receiver performs some method to carry out the request. Every message may include any additional information as arguments.**

Responsibilities

- In object-oriented programming, behaviors of an object described in terms of responsibilities.
- In our example, my request for action indicates only the desired outcome (food delivered to my family). The agent (zomato) free to use any technique that solves my problem.

Classes and Instances

- In object-oriented programming, all objects are instances of a class. The method invoked by an object in response to a message is decided by the class. All the objects of a class use the same method in response to a similar message.
- In our example, the zomato a class and all the hotels are sub-classes of it. For every request (message), the class creates an instance of it and uses a suitable method to solve the problem.

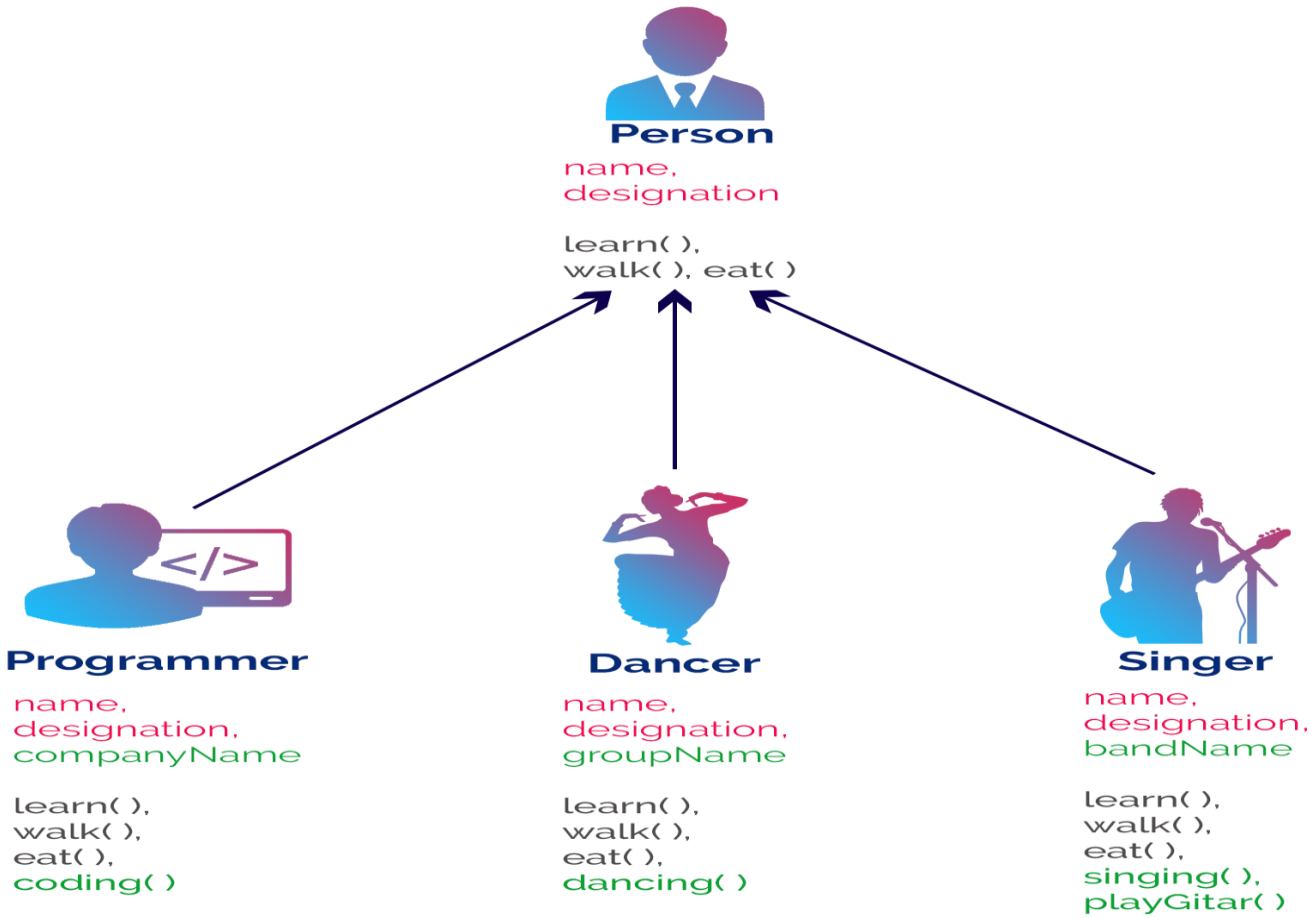
OOP concepts in JAVA

- OOP stands for Object-Oriented Programming.
- OOP is a programming paradigm in which every program is follows the concept of object. In other words, OOP is a way of writing programs based on the object concept.
- The object-oriented programming paradigm has the following core concepts:
  - Encapsulation
  - Inheritance
  - Polymorphism
  - Abstraction

Encapsulation

- Encapsulation is the process of combining data and code into a single unit (object / class)
- In OOP, every object is associated with its data and code.
- In programming, data is defined as variables and code is defined as methods.
- The java programming language uses the class concept to implement encapsulation.

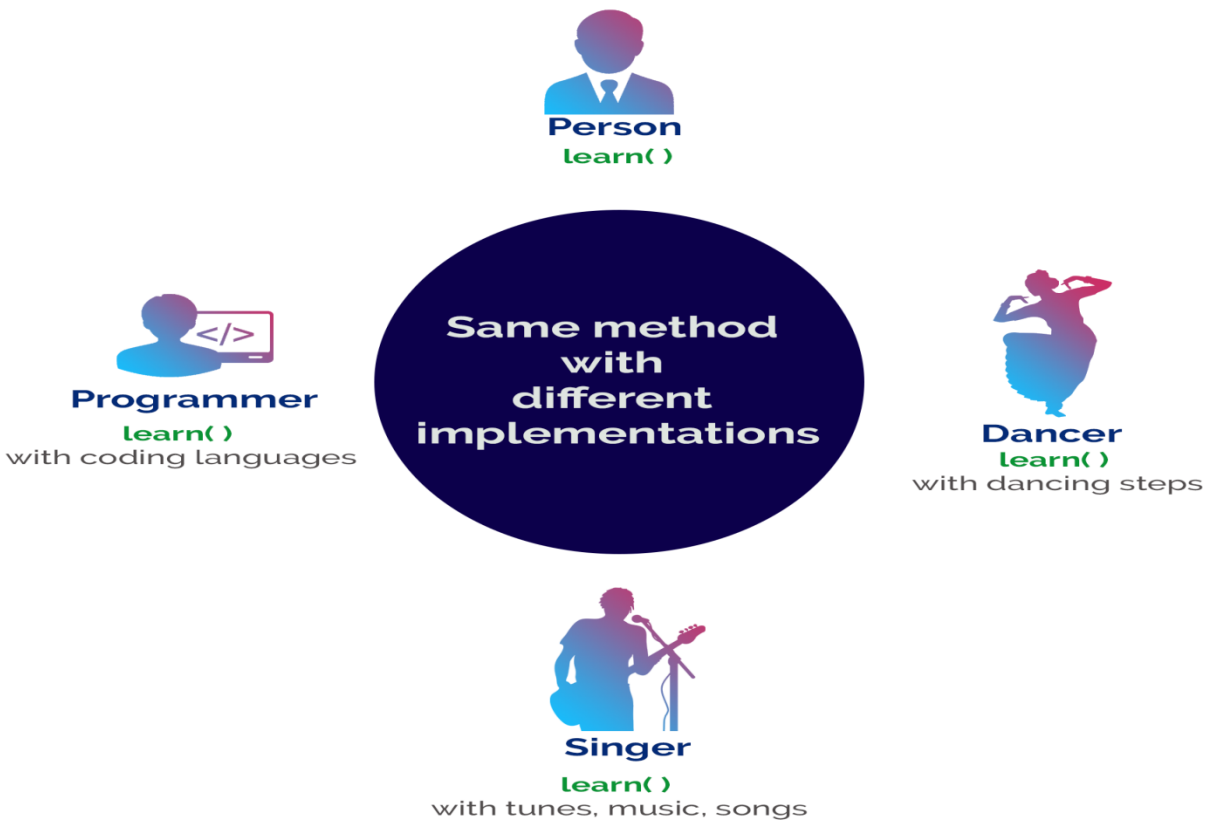
Inheritance



Continue…

- Inheritance is the process of acquiring properties and behaviors from one object to another object or one class to another class.
- In the inheritance concept, the class which provides properties is called as parent class and the class which receives the properties is called as child class.
- The parent class is also known as base class or super class. The child class is also known as derived class or sub class.
- The properties and behaviors of base class extended to its derived class, but the base class never receive properties or behaviors from its derived class.

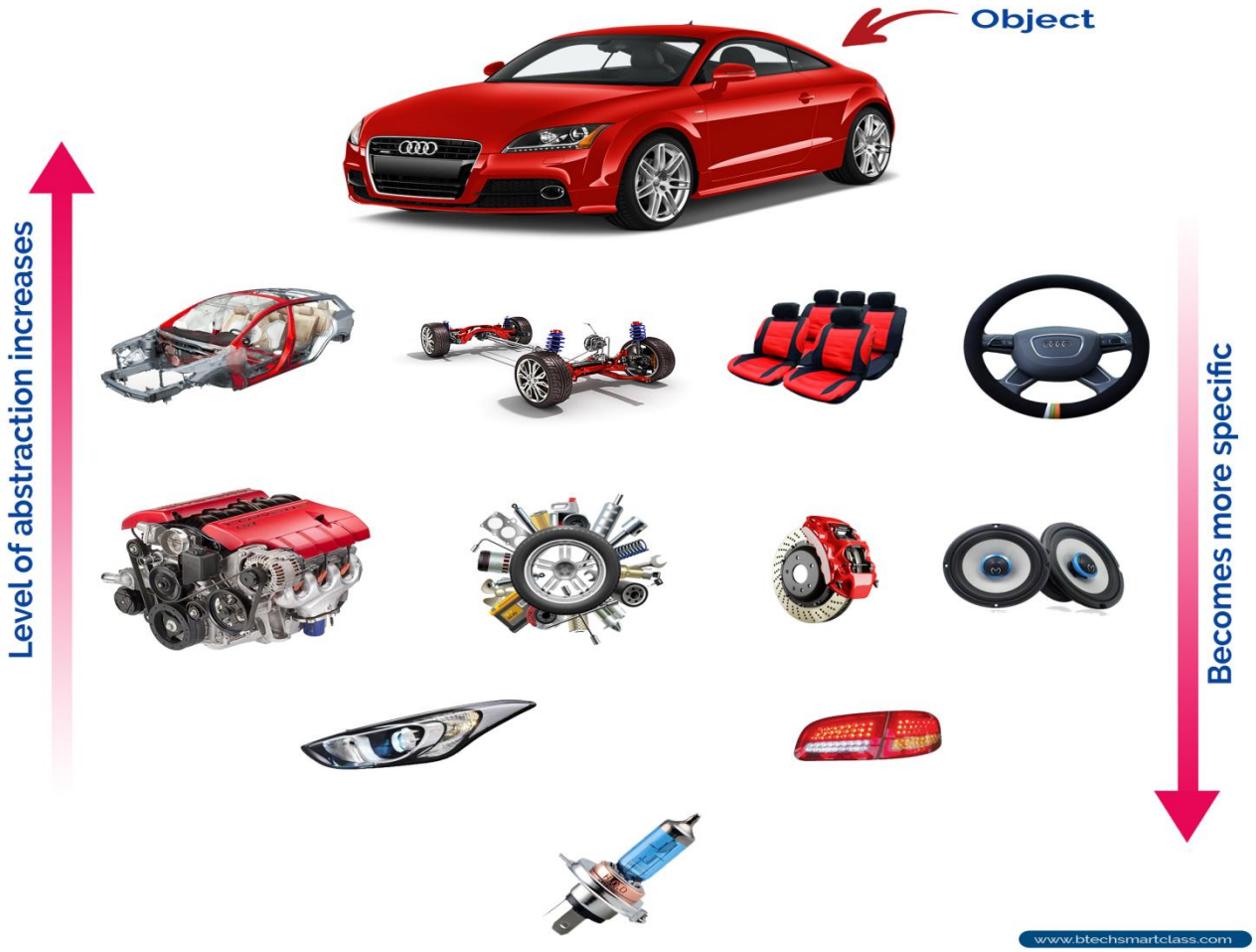
Polymorphism



Continue…

- Polymorphism is the process of defining same method with different implementation.
- That means creating multiple methods with different behaviors.
- The java uses method overloading and method overriding to implement polymorphism.
- Method overloading - multiple methods with same name but different parameters.
- Method overriding - multiple methods with same name and same parameters.

Abstraction



Continue…

- Abstraction is hiding the internal details and showing only essential functionality.
- In the abstraction concept, we do not show the actual implementation to the end user, instead we provide only essential things.
- For example, if we want to drive a car, we does not need to know about the internal functionality like how wheel system works? how brake system works? how music system works? etc.

Java Buzz Words

- Java is the most popular object-oriented programming language.
- Java has many advanced features, a list of key features is known as Java Buzz Words.

- **Simple**
- **Secure**
- **Portable**
- **Object-oriented**
- **Robust**
- **Architecture-neutral (or) Platform Independent**
- **Multi-threaded**
- **Interpreted**
- **High performance**
- **Distributed**
- **Dynamic**

Contd…

- **Simple:** Java programming language is very simple and easy to learn, understand, and code. Most of the syntaxes in java follow basic programming language C and object-oriented programming concepts are similar to C++. In a java programming language, many complicated features like pointers, operator overloading, structures, unions, etc. have been removed. One of the most useful features is the garbage collector it makes java more simple.
- **Secure:** Java is said to be more secure programming language because it does not have pointers concept, java provides a feature "applet" which can be embedded into a web application. The applet in java does not allow access to other parts of the computer, which keeps away from harmful programs like viruses and unauthorized access.

Contd…

- **Portable:** Portability is one of the core features of java which enables the java programs to run on any computer or operating system. For example, an applet developed using java runs on a wide variety of CPUs, operating systems, and browsers connected to the Internet.
- **Object-oriented:** Java is said to be a pure object-oriented programming language. In java, everything is an object. It supports all the features of the object-oriented programming paradigm. The primitive data types java also implemented as objects using wrapper classes, but still, it allows primitive data types to archive high-performance.

Contd…

- **Robust:** Java is more robust because the java code can be executed on a variety of environments, java has a strong memory management mechanism (garbage collector), java is a strictly typed language, it has a strong set of exception handling mechanism, and many more.
- **Platform Independent:** Java has invented to archive "write once; run anywhere, any time, forever". The java provides JVM (Java Virtual Machine) to to archive architectural-neutral or platform-independent. The JVM allows the java program created using one operating system can be executed on any other operating system.

Contd…

- **Multi-threaded:** Java supports multi-threading programming, which allows us to write programs that do multiple operations simultaneously.
- **Interpreted:** Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java byte code. The byte code is interpreted to any machine code so that it runs on the native machine.
- **High performance:** Java provides high performance with the help of features like JVM, interpretation, and its simplicity.

Contd…

- **Distributed:** Java programming language supports TCP/IP protocols which enable the java to support the distributed environment of the Internet. Java also supports Remote Method Invocation (RMI), this feature enables a program to invoke methods across a network.
- **Dynamic:** Java is said to be dynamic because the java byte code may be dynamically updated on a running system and it has a dynamic memory allocation and de-allocation (objects and garbage collector).

Java Overview

- Java is a computer programming language.
- Java was created based on C and C++. Java uses C syntax and many of the object-oriented features are taken from C++.
- Before Java was invented there were other languages like COBOL, FORTRAN, C, C++, Small Talk, etc.
- Java was invented by a team of 13 employees of Sun Microsystems, Inc. which is lead by James Gosling, in 1991.
- Java was developed as a part of the Green project. Initially, it was called Oak, later it was changed to Java in 1995.

History

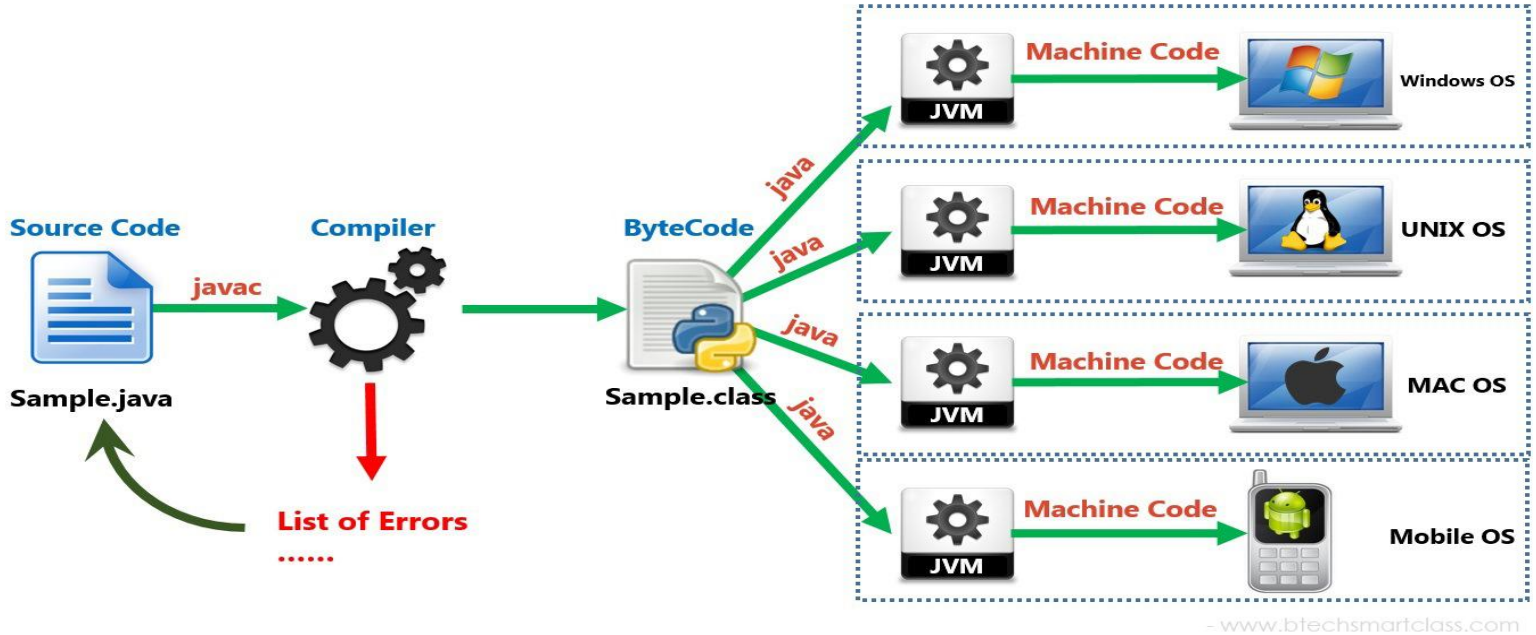
- The C language developed in 1972 by Dennis Ritchie had taken a decade to become the most popular language.
- In 1979, Bjarne Stroustrup developed C++, an enhancement to the C language with included OOP fundamentals and features.
- A project named “Green” was initiated in December of 1990, whose aim was to create a programming tool that could render obsolete the C and C++ programming languages.
- Finally in the year of 1991 the Green Team was created a new Programming language named “OAK”.
- After some time they found that there is already a programming language with the name “OAK”.

Contd…

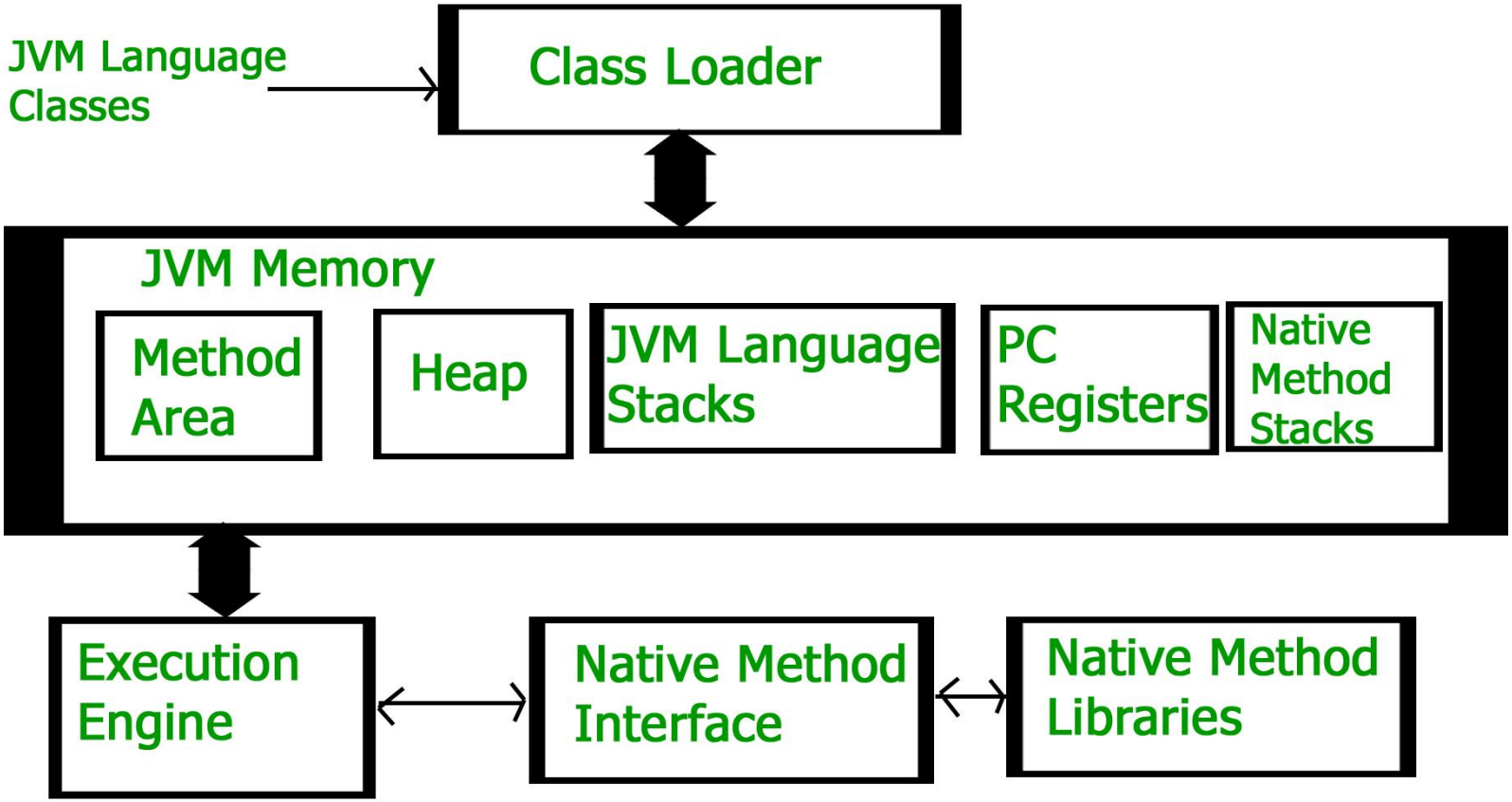
- Why Java named "Oak"?

- Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.
- So, the green team had a meeting to choose a new name. After so many discussions they want to have a coffee. They went to a Coffee Shop which is just outside of the Gosling's office and there they have decided name as "JAVA".
- Java is an island of Indonesia where the first coffee was produced (called java coffee). It is a kind of espresso bean.

Execution Process of Java Program



- Create a source code (.java file).
- Compile the source code using javac command.
- Run or execute .class file uisng java command.


JVM Language Classes
Class Loader
JVM Memory
Method Area
Heap
JVM Language Stacks
PC Registers
Native Method Stacks
Execution Engine
Native Method Interface
Native Method Libraries

Sample Program

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
         System.out.println (“ Welcome to Java…. ”);
   }
}
```

The class Keyword is used to declare a class. Keywords are reserved words that have a special meaning. Here, the class keyword defines the class Welcome. The braces { } known as delimiters, are used to indicate the start and end of a class body

Sample Program

```
class   Welcome  {
    public  static  void  main( String args [ ] ) {
            System.out.println (“ Welcome to Java…. ”);
    }
}
```

The Class Name. Welcome is the name of the class defined by using the **`class`** keyword.

Sample Program

```
class   Welcome  {
    public  static  void  main( String args [ ] ) {
            System.out.println (“ Welcome to Java…. ”);
    }
}
```

Naming Conventions

- Should be meaningful (Strongly recommended)
- First letter of each word of class should be caps.

Naming Conventions in Java

The following naming conventions are often followed, although not enforced by Java:

- Names of ***classes*** and ***interfaces*** begin with an uppercase letter. (Pascal Case)

  Examples : Student , EmployeeData

- Names of ***methods*** and ***variables*** follows camel Case, i.e., If a name is composed of several words, then each word (except possibly the first one) begins with an uppercase letter.

  Examples : setDetails ( ) , showResult ( )

  vehicle , myVehicle

- Named ***constants*** (that is, final variables and fields) are written entirely in uppercase, and the parts of composite names are separated by underscores (_).

  Examples : CENTER , MAX_VALUE.

- Names of ***packages*** written entirely in lowercase.

  Examples : java.lang , java.awt.event

For uniqueness, they are often prefixed with reverse domain names, as in com. sun.xml.util.

Sample Program

```
class   Welcome  {
    public  static  void  main( String args [ ] ) {
            System.out.println (“ Welcome to Java…. ”);
    }
}
```

Naming Rules in Java

- A name must begin with a letter, an underscore (_), or the dollar symbol ($), which can be followed by a sequence of letters or digits (0 to 9), ‘$’, or ‘_’ .
- A name should not start with a digit.
- A name should not contain embedded white spaces .
- A name should not consist of a keyword.
- A name in Java is case sensitive.

Sample Program

```
class   Welcome  {

    public  static  void  main( String args [ ] ) {

        System.out.println (“ Welcome to Java…. ”);

    }

}
```

Access Specifier

Return type

Parameters for Command-line arguments

Access Modifier

Method Name

Sample Program

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
         System.out.println (“ Welcome to Java…. ”);
   }
}
```

An access specifier defines the scope of a class member. A class member refers to the variables and methods in a class.

Java supports the following access specifiers:

- public
- private
- protected
- default

Sample Program

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
         System.out.println (“ Welcome to Java…. ”);
   }
}
```

Here, main method is declared **public**. Any member that is declared public can be accessed from outside the class with object.

Sample Program

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
         System.out.println (“ Welcome to Java…. ”);
   }
}
```

The keyword **`static`** is an access modifier. Here, static keyword describes main method does not need an object to get invoked.

Sample Program

```
class   Welcome  {
    public  static  void  main( String args [ ] ) {
            System.out.println (“ Welcome to Java…. ”);
    }
}
```

The keyword **void** is a return type. It means main method does not return any thing after execution.

Sample Program

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
         System.out.println (“ Welcome to Java…. ”);
   }
}
```

The main method

The first line of code that a java compiler looks for in the source file is the main method. This function is the entry point of the application.

The **`main method`** is ideally used to create objects and invoke methods.

main( ) is the first method which is executed in a java program.

Sample Program

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
         System.out.println (“ Welcome to Java…. ”);
   }
}
```

Parameters for command line arguments. Values passed during the execution of program through command line goes in the form of strings. Hence, string array variable **`String args[ ]`** is declared in main method to hold the values.

Sample Program

```
class   Welcome  {
   public  static  void  main( String args [ ] ) {
         System.out.println (“ Welcome to Java…. ”);
   }
}
```

System is a class that belongs to the **lang** package. A package is a collection of classes. The System class contains the methods **print( ) and println( )**, which displays the enclosed text on the screen. The character **(.)** is used to access the method, which coded in the System class.

Variables

- A variable is a location in the memory that has a name and contains a value. The value could be integer, float, or a character.

Naming Variables in Java

The following rules are used for naming variables in Java :

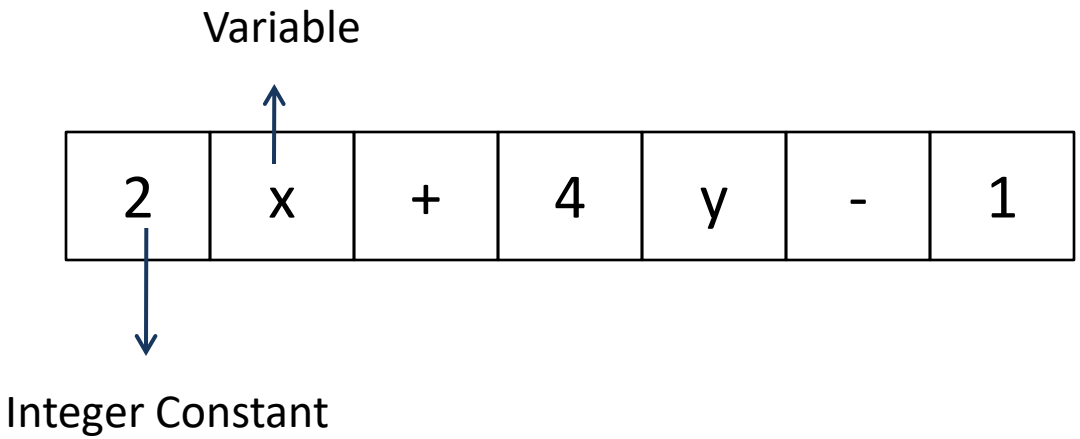
- A variable name must begin with a letter, an underscore (_), or the dollar symbol ($), which can be followed by a sequence of letters or digits (0 to 9), ‘$’, or ‘_’ .
- Should not contain any embedded spaces or symbols. However, an underscore can be used wherever a space is required, like high_score.
- Must be unique.
- Uppercase letters are considered distinct from lowercase letters
- Can have any number of characters.
- Reserved Keywords cannot be used as variable names.

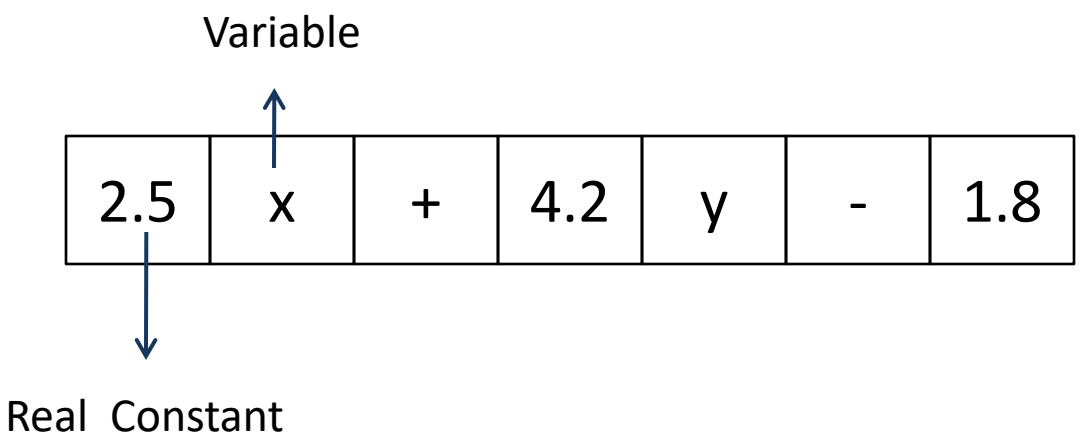
Instructions

2	x	+	4	y	-	1

2.5	x	+	4.2	y	-	1.8

Instructions





Data Types in Java

- Data type specifies the size and type of values that can be stored in a variable.
- A variable is associated with a data type.
- Data types in Java are classified into two types:
  - Primitive — which include Integer, Character, Boolean, and Floating Point.
  - Non-primitive (Reference) — which include Classes, Interfaces, and Arrays.

Primitive Data Types

Integer

Type	Size (in bytes)	Range	Default Value
byte	1	-128 to 127	0
short	2	-32768 to 32767	0
int	4	-2147483648 to 2147483647	0
long	8	-9223372036854775808 to 9223372036854775807	0L

Primitive Data Types

Floating Point

Type	Size (in bytes)	Range	Default Value
float	4	3.4e−038 to 3.4e+038	0.0f
double	8	1.7e−308 to 1.7e+038	0.0d

Primitive Data Types

Character

- The char type is a Unicode character, as defined by the Unicode Standard
- It has a minimum value of ‘\u0000’ (or 0) and a maximum value of ‘\uffff’ (or 65,535, inclusive).

Type	Size (in bytes)	Range	Default Value
char	2	0 to 65535	'\u0000'

Boolean

Type	Size (in bytes)	Range	Default Value
boolean	1	true or false	false

Non-Primitive Data Types

The String type in Java

- A String is a sequence of text characters. You typically create a string with a String literal, enclosed in quotes: "This is an example of a string."

Type	Default Value
String	null

Escape Sequences in Java

Java Character Escape Sequences

- Some characters aren't printable, but you still need to use them in strings.

Escape Sequence	Meaning	Escape Sequence	Meaning
\'	Single Quote	\f	form Feed
\"	Double Quote	\n	Newline
\\	Backslash	\r	Carriage Return
\0	Null, not the same as the C# null value	\t	Horizontal Tab
\a	Bell	\v	Vertical Tab
\b	Backspace		

Java Operators

- Operators are used to compute results and compare the data values of a program.
- Applications use operators to process the data entered by a user.
- Operators can transform one or more data values, called operands, into a new data value.

Operators in Java

- Arithmetic operators
- Arithmetic assignment operators
- Unary operators
- Comparison operators
- Logical operators
- Conditional operator

Arithmetic Operators

- These operators are the symbols that are used to perform arithmetic operations on variables.

Operator	Description
+	Used to add two numbers
-	Used to subtract two numbers
*	Used to multiply two numbers
/	Used to divide one number by another
%	Used to divide two numbers and return the remainder.

Arithmetic operators

Integer Arithmetic

```
Assume  a = 14 , b = 4
        a + b  = 18
        a – b  =  10
        a * b  =  56
        a / b   = 3  ( Decimal part truncated )
        a % b = 2  ( Remainder )
```

Real Arithmetic

```
Assume a = 20.5 ,  b = 6.4
        a + b  = 26.9
        a – b  = 14.1
        a * b  = 131.2
        a / b   = 3.20313
        a % b = 1.29999
```

Mixed - Mode Arithmetic

Mixed Mode Arithmetic
int / int = int
int / double = double
double / double = double
double / int = double

```
15.0 / 10 = 1
15 / 10.0 = 1.5
```

```
-14 % 3 = -2
 14 % -3 = 2
```

During Modulo division the sign of the result is always the sign of the first operand.

Arithmetic Assignment Operators

- These operators are used to perform arithmetic operations to assign a value to an operand.
- The simplest of these is the **“ = “**.

Operator	**Usage**	**Description**
=	x = 5	Stores the value 5 in the variable x
+=	x + = y	Same as: x = x + y
-=	x - = y	Same as: x = x – y
*=	x * = y	Same as: x = x * y
/=	x / = y	Same as: x = x / y
% =	x % = y	Same as: x = x % y

Arithmetic Assignment operators

Unary Operators

- These operators are used to increment or decrement the value of an operand by 1.

Operator	Usage	Description
++	x++ (post increment) ++x (pre increment)	Used to increment the value of an operand by 1.
--	x-- (post decrement) --x (pre decrement)	Used to decrement the value of an operand by 1.

Unary Operators

Comparison Operators

- These operators are used to compare two values and perform an action on the basis of the result of that comparison.
- The comparison operator returns a boolean value ‘true’ or ‘false’

Operator	Usage	Description
= =	x = = y	Used to check whether the value of x is equal to the value of y.
>	x > y	Used to check whether the value of x is greater than the value of y.
<	x < y	Used to check whether the value of x is less than the value of y.
> =	x > = y	Used to check whether the value of x is greater than or equal to the value of y.
< =	x < = y	Used to check whether the value of x is less than or equal to the value of y.
! =	x ! = y	Used to check whether the value of x is not equal to the value of y.

Comparison Operators

Logical Operators

- These operators are used to evaluate expressions and return a boolean value ‘true’ or ‘false’

Operator	Usage	Description
&&	expr1 && expr2	Returns true if both expression1 and expression2 are true.
||	expr1 || expr2	Returns true if either expression1 or expression2 or both of them are true.
!	! expression	Returns true if the expression is false.
^	expr1 ^ expr2	Returns true if either expression1 or expression2 is true. It returns false if both expression1 and expression2 are true of if both are false.

Logical Operators

Conditional Operator

- The conditional operator (**?:**) returns one of two values depending on the value of a Boolean expression.

- Syntax:

```
(condition) ? true_expression : false_expression;
```

- Example: sets the absolute value in the variable x

```
int x = <value>;
 x = (x > 0) ? x : -x;
```

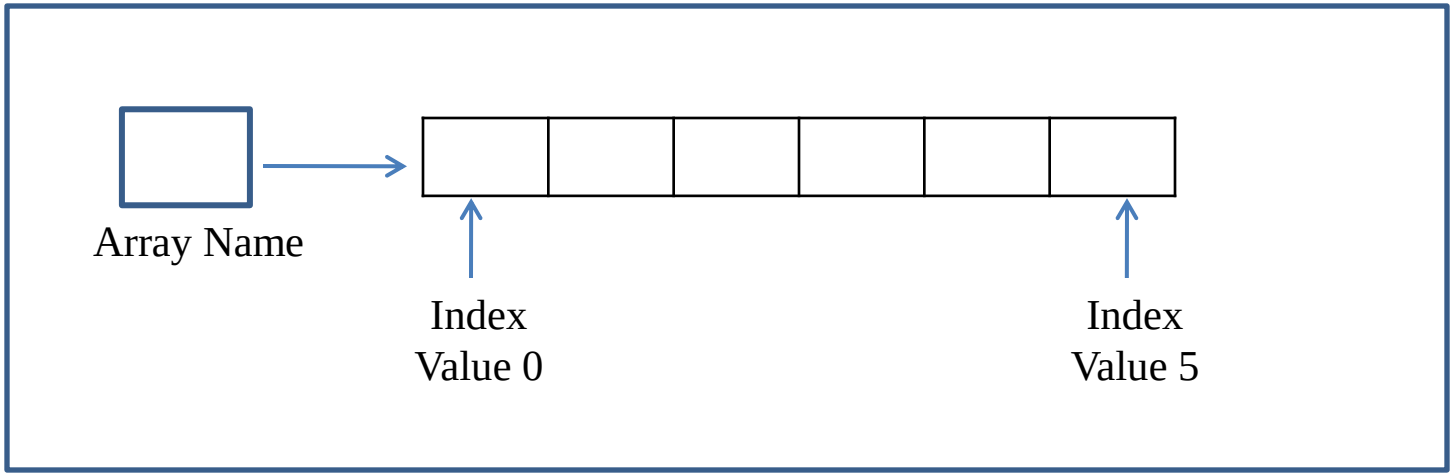
Java Operators Precedence and Associativity

- The following table describes the allowable operators, their precedence, and associativity.

Category (by precedence)	Operator(s)	Associativity
Unary	+ - ! ~ ++x --x (T)x	right
Multiplicative	* / %	left
Additive	+ -	left
Shift	<< >>	left
Relational	< > <= >= is as	left
Equality	== !=	left
Logical AND	&	left
Logical XOR	^	left
Logical OR	|	left
Conditional AND	&&	left
Conditional OR	||	left
Null Coalescing	??	left
Ternary	?:	right
Assignment	= *= /= %= += -= <<= >>= &= ^= |= =>	right

Implementing Arrays

- An **`array`** is a collection of values of the same data type.
- The values in an array are called the array elements.
- Array is a reference / non primitive data type.



Implementing Arrays

Declaring an Array

```
datatype  Arrayname [ ] ;

int score [ ];   (or)  int [ ] score ;
```

Initializing an Array

- Array is a reference type, therefore you need to use the new keyword to create an instance of the array.

```
int score [ ] ;                // Array declaration
score = new int[10];           // Array Instance
```

- The preceding two statements can be combined into a single statement

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

Implementing Arrays

Assigning Values to the Array

```
int score [ ] = new int[10];
score [0] = 5;
score [1] = 8;

int score [ ] = {5,10,15};
```

length property

```
int x [ ] = new int [8];

x.length  → returns size of the array
            Ans:8
```

Implementing Arrays

foreach Usage

- A statement that iterates through an array and executes the same set of instructions on each element is very common. The **foreach** statement interprets the common looping process by removing the need for you to check the array size.

```
for ( type identifier : expression )
```

- The following code snippet shows the usage of foreach statement:

```
int num [ ] = {4,6,2,8,1,-3,9,-5,4,7};
for( int k : num ){
        System.out.println(k);
}
```

Double Dimensional Array

Declaring a 2D Array

```
datatype  Arrayname [ ][ ];

int  p [ ][ ];   (or)  int[ ][ ]  p;
```

Initializing a 2D Array

```
int p [ ][ ];          // Array declaration
p = new int[2][3];     // Array Instance
```

- The preceding two statements can be combined into a single statement

```
int p [ ][ ] = new int[2][3];
```

Implementing Arrays

length property

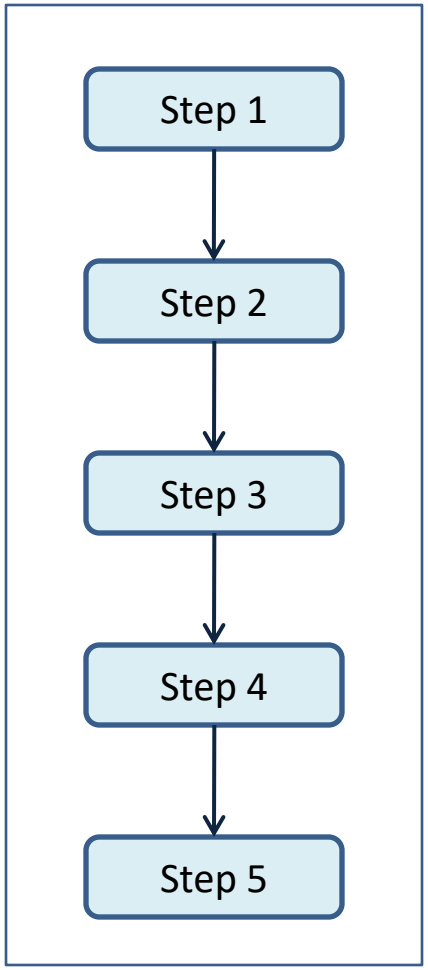
```
int x [ ][ ] = new int [2][3];

x.length  → returns no. of rows
            Ans:2
```

foreach Usage of 2D Array

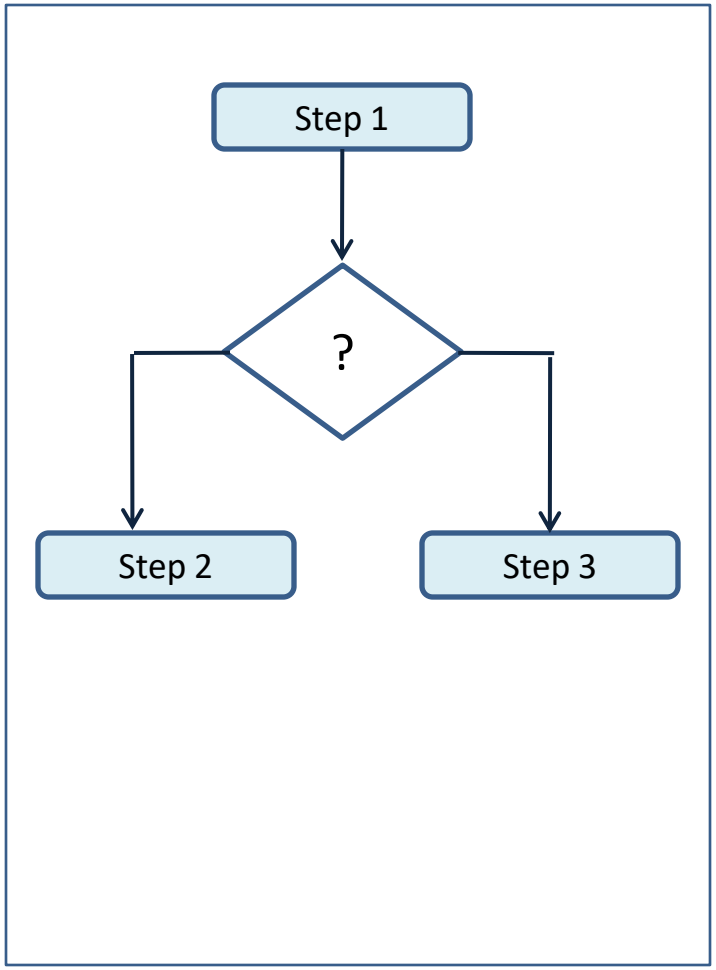
- The following code snippet shows the usage of foreach statement:

```
int num [ ][ ] = {{1,2,3},{4,5,6}};
for( int i[ ] : num ){
        for( int j : i ) {
                System.out.println( j );
        }
}
```

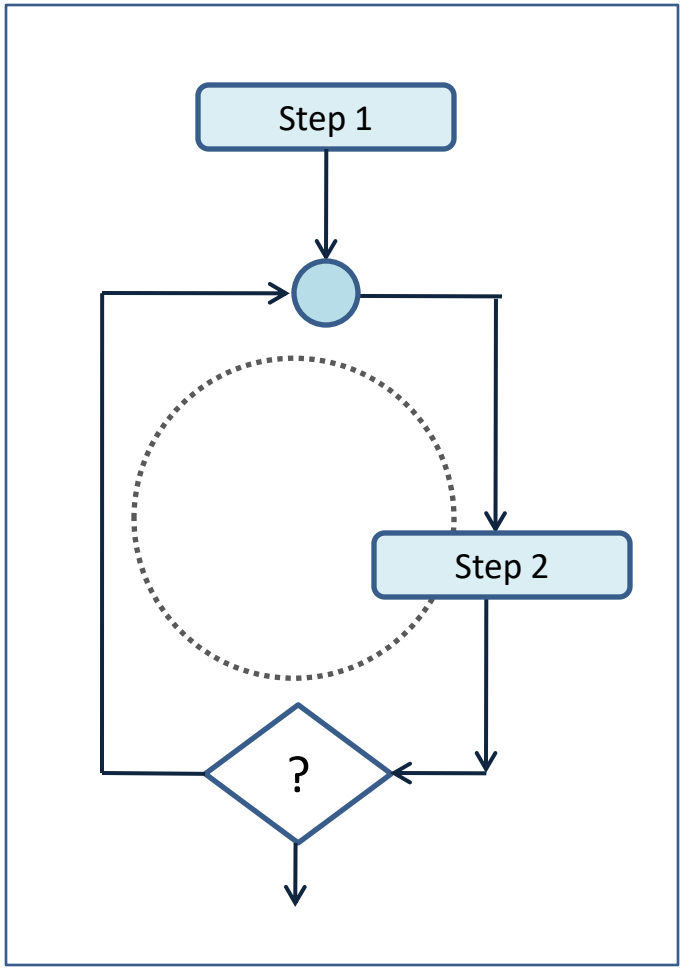
Programming Styles



Sequence Style



Selection Style

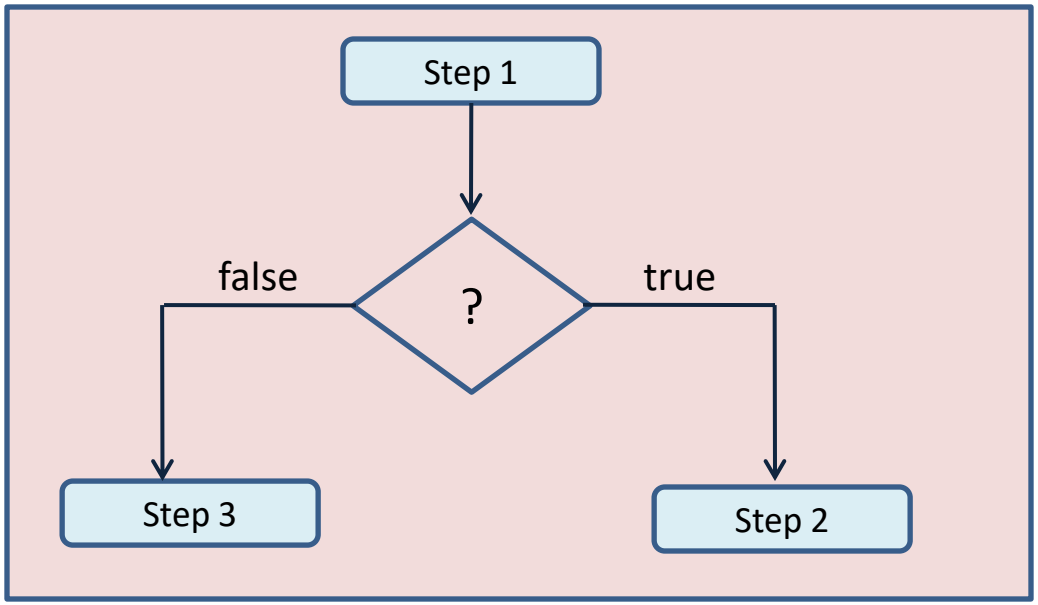


Looping Style

Introduction

We have seen that a program is a set of statements, which are normally executed sequentially in the order in which they appear. This happens when no options or no decision on certain calculations are necessary.

However, in practice, we have a number of situations where we may have to change the order of execution of statements based on certain conditions. This involves a kind of decision making to see whether a particular condition, has occurred or not and direct the computer to execute certain statements accordingly.



Selection Style

Introduction

When a program breaks the sequential flow and jumps to another part of the code, it is called **Branching**.

When the branching is based on a particular condition, it is known as **Conditional Branching**.

If branching takes place with out any decision, it is known as **Unconditional Branching**.

Decision Making Control Statements

Java Language supports the following statements known as Decision Making Control Statements.

- If Statement
- Switch Statement

If Statement

The If statement is a powerful decision making statement and it is used to control the flow of execution of statements.

It is basically a two – way decision statement.

- True Way
- False way

Syntax :

```
if ( condition )
```

It allows the compiler to evaluate the condition first and returns the boolean value (true / false)

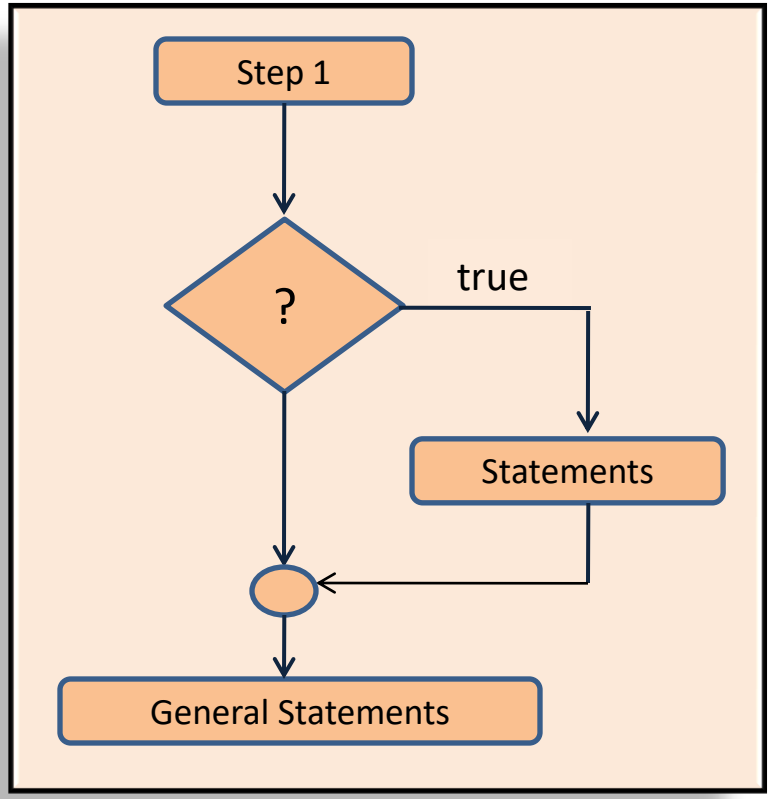
Different forms of If Statement

The if statement can be implemented in different forms depending on the complexity of conditions to be tested.

- Simple if Statement
- if.. else Statement
- else if Ladder
- Nested if.. else Statement

Simple **if** Statement

```
if ( condition )  {
        Statements-block;
}
General Statements;
```

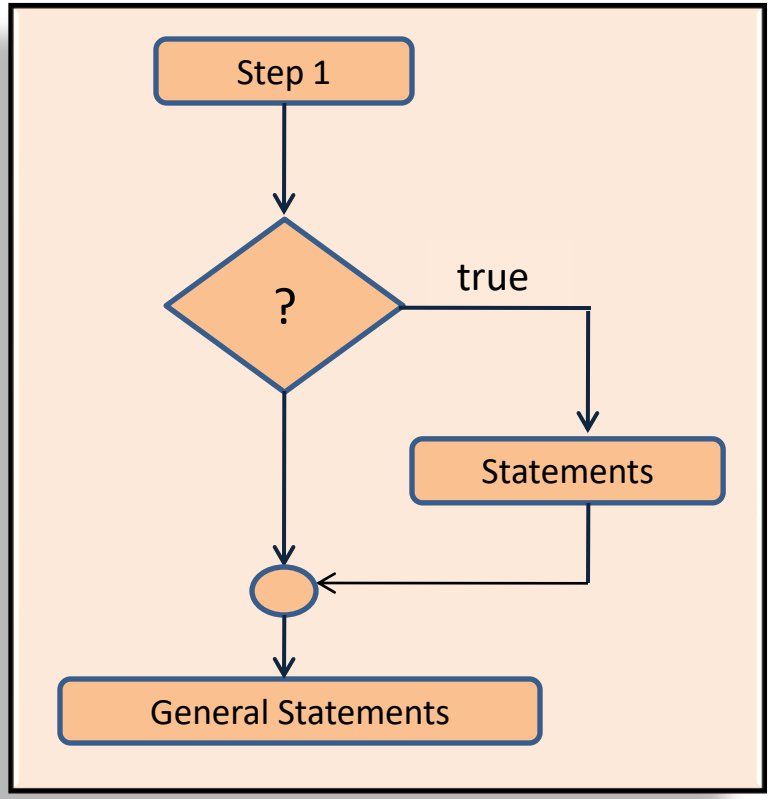


The statements-block may be a single statement or group of statements. When there is only a single statement then no need to put **{ }** braces, but it is a good programming practice to put braces.

If the given condition is **true**, the statements-block will be executed, otherwise the statements-block will be skipped and execution will jump to the general statements.

Simple **if** Statement

```
if ( condition )   {
            Statements-block;
}
General Statements;
```

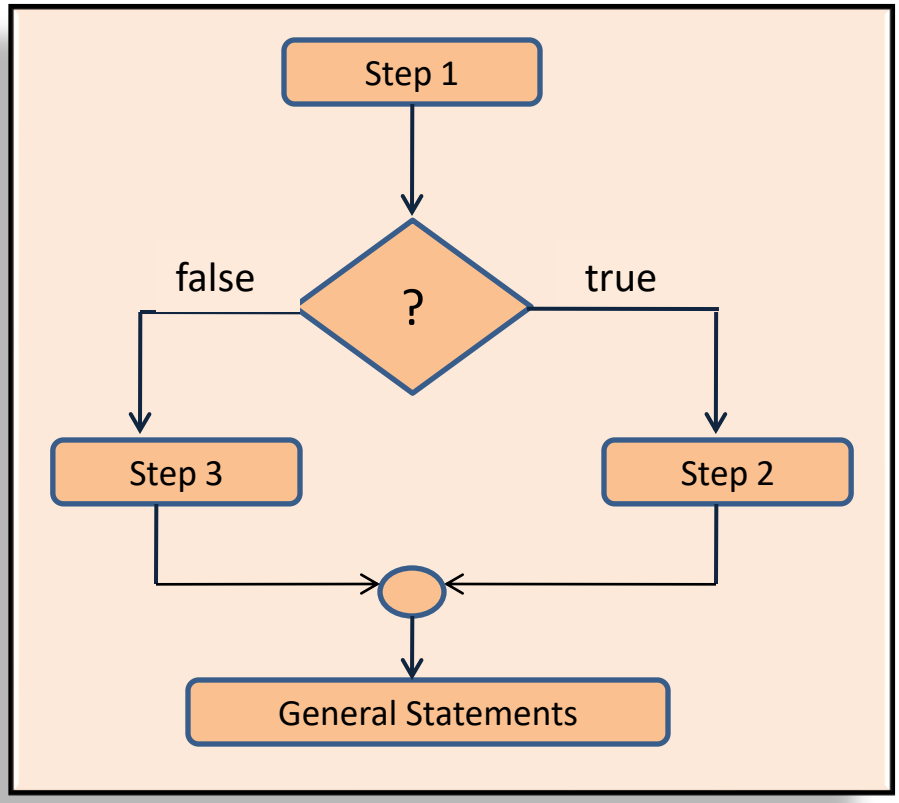


for example :

```
if ( category = = SPORTS  )  {
                marks = marks+bonus;
}
print  marks ;
```

if.. else Statement

```
if ( condition )  {
        Statements-block;
}
else  {
        Statements-block;
}
```



else if Ladder

When there are series of decisions.

```
if ( condition 1 )   {
            Statements-block;
}
else if ( condition 2 )   {
            Statements-block;
}
else if ( condition 3 )   {
            Statements-block;
}
else   {
            Statements-block;
}
```

Nested if.. else When the decisions are based on another decisions.

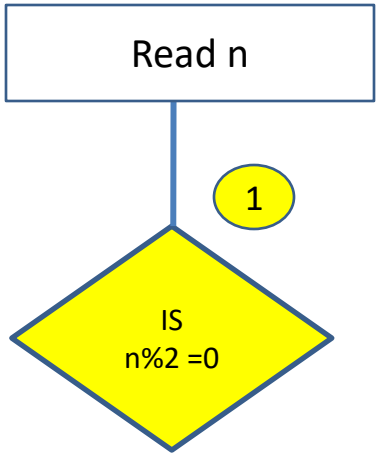
```
if ( condition 1 )  {
        if ( condition )  {
                Statements-block;
        }
        else  {
                Statements-block;
        }
}
else  {
        if ( condition )  {
                Statements-block;
        }
        else  {
                Statements-block;
        }
}
```

- Find out the given number is an even or odd

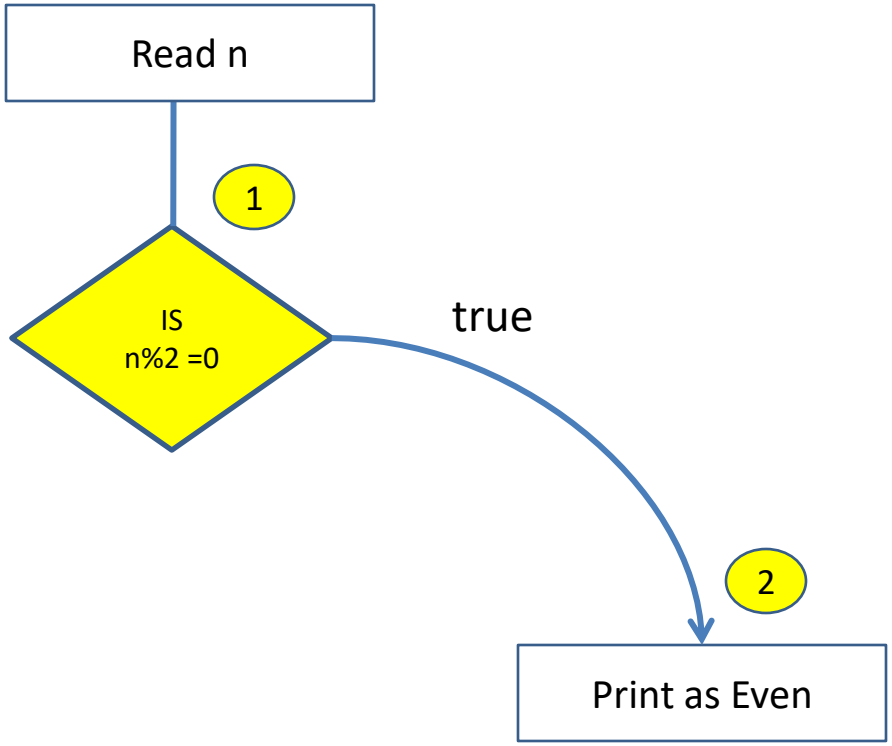
- Find out the given number is an even or odd

Read n

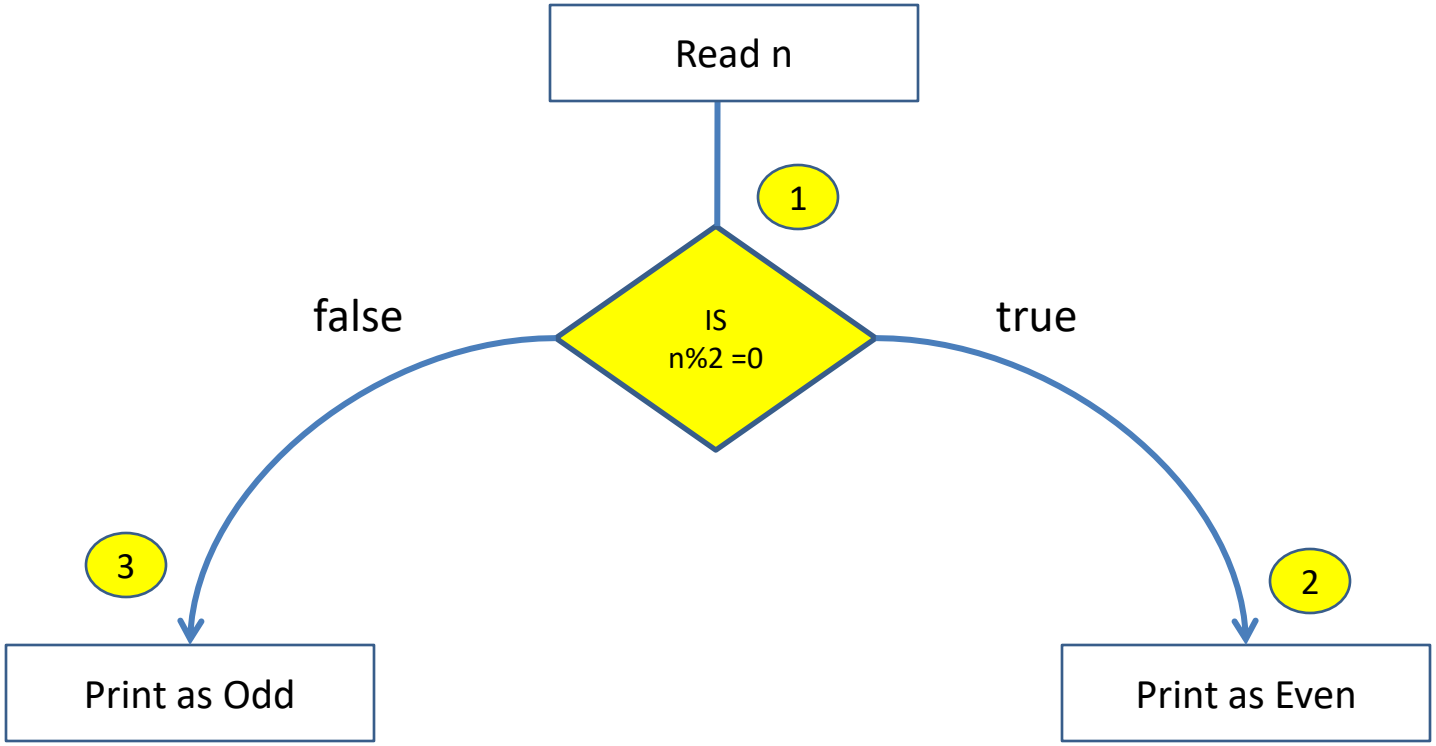
- Find out the given number is an even or odd



- Find out the given number is an even or odd

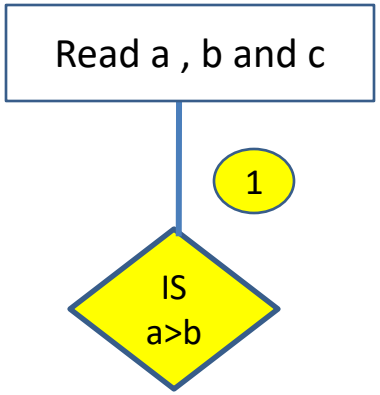


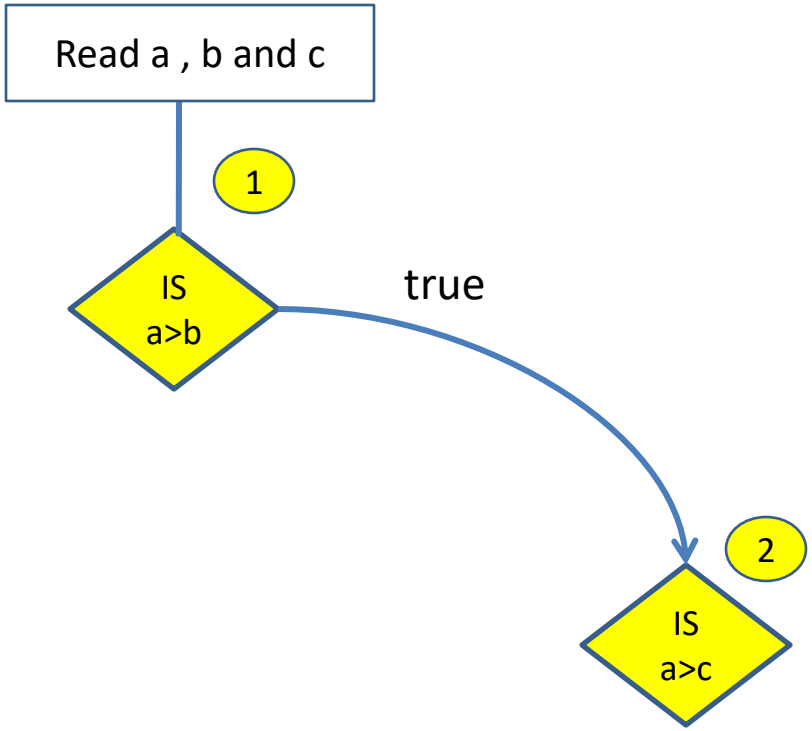
- Find out the given number is an even or odd

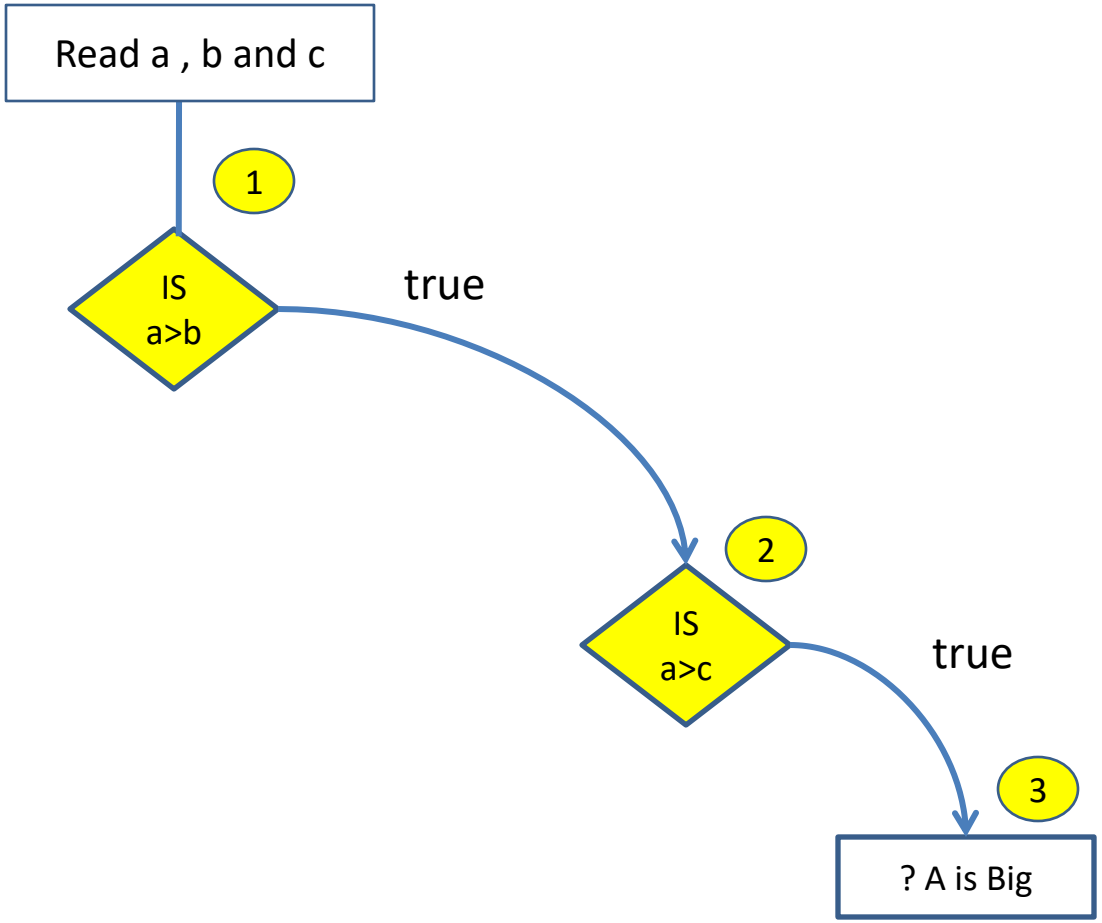


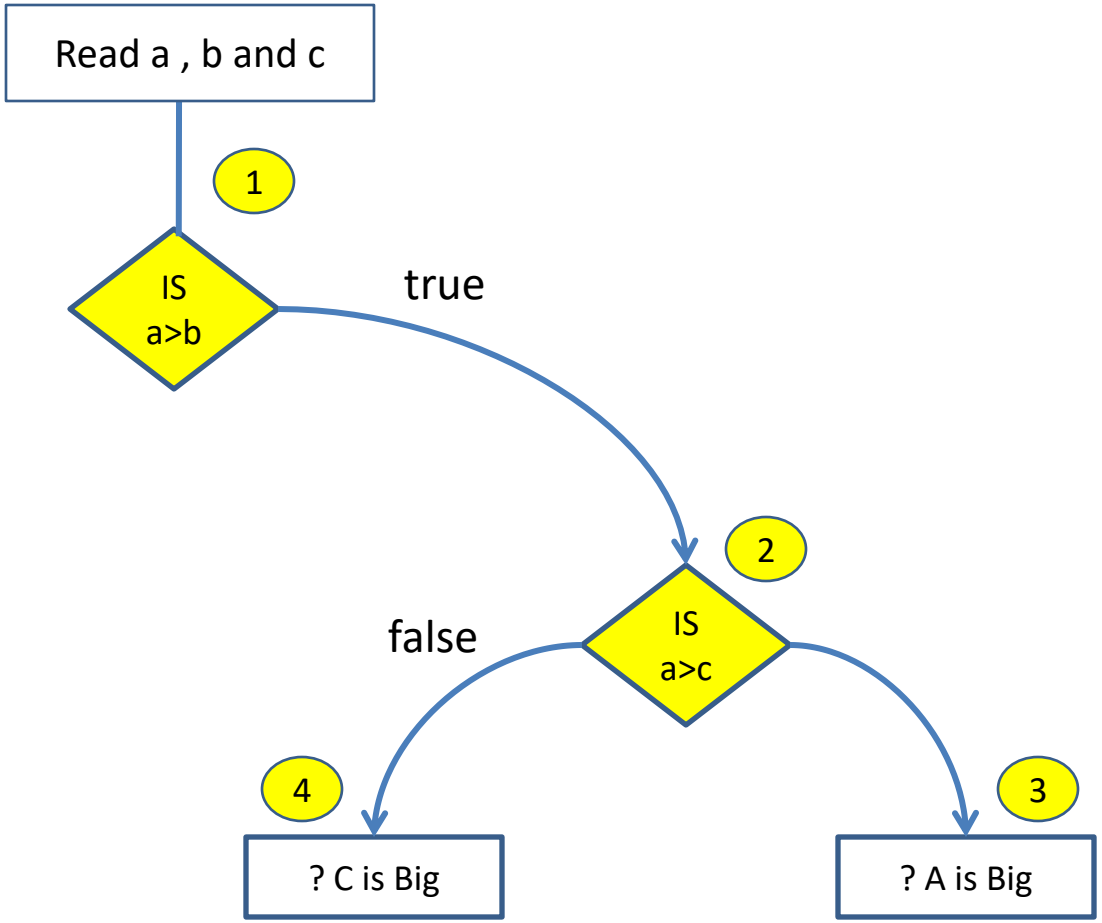
- Find out the big number in the given 3 unique numbers

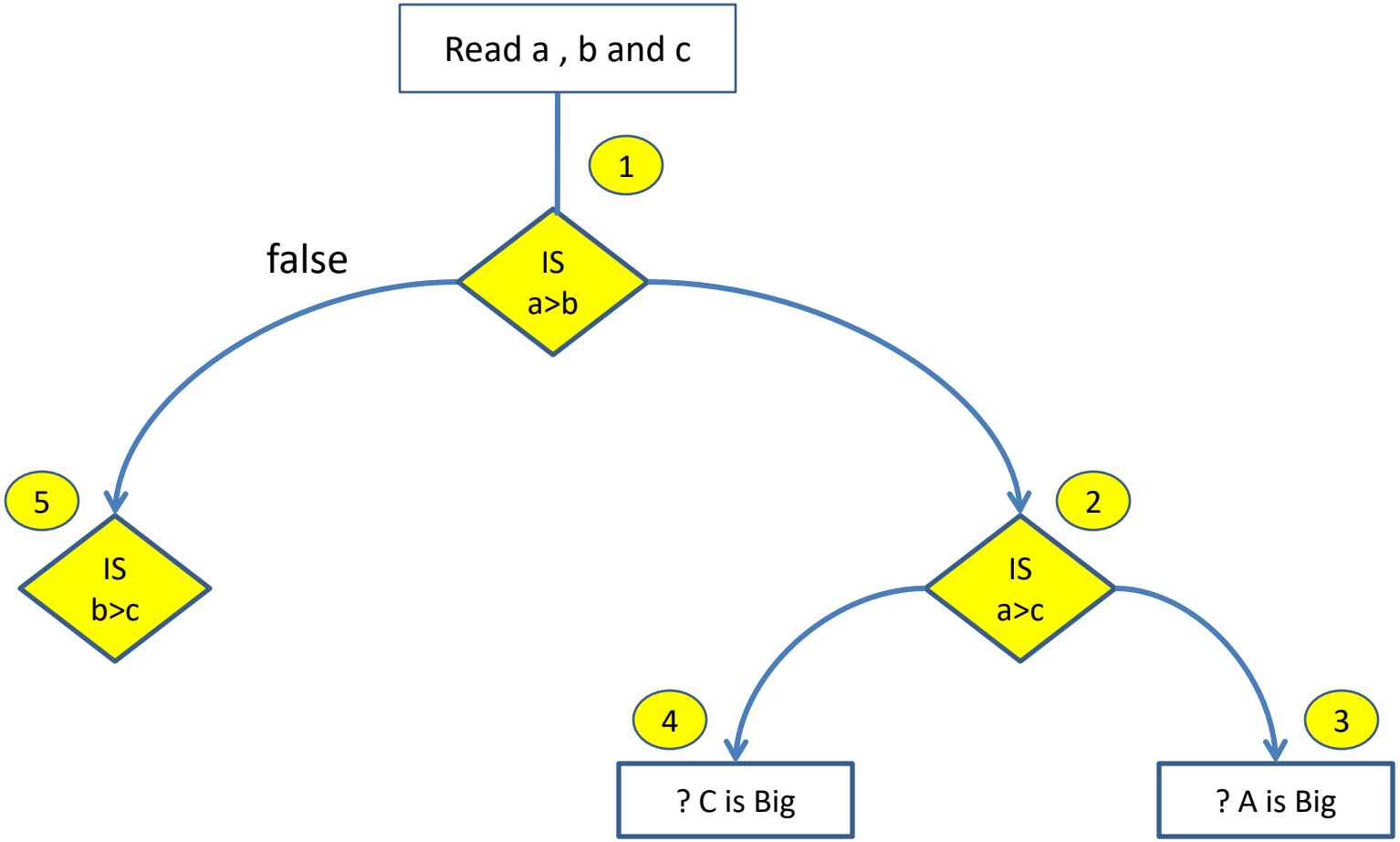
Read a , b and c

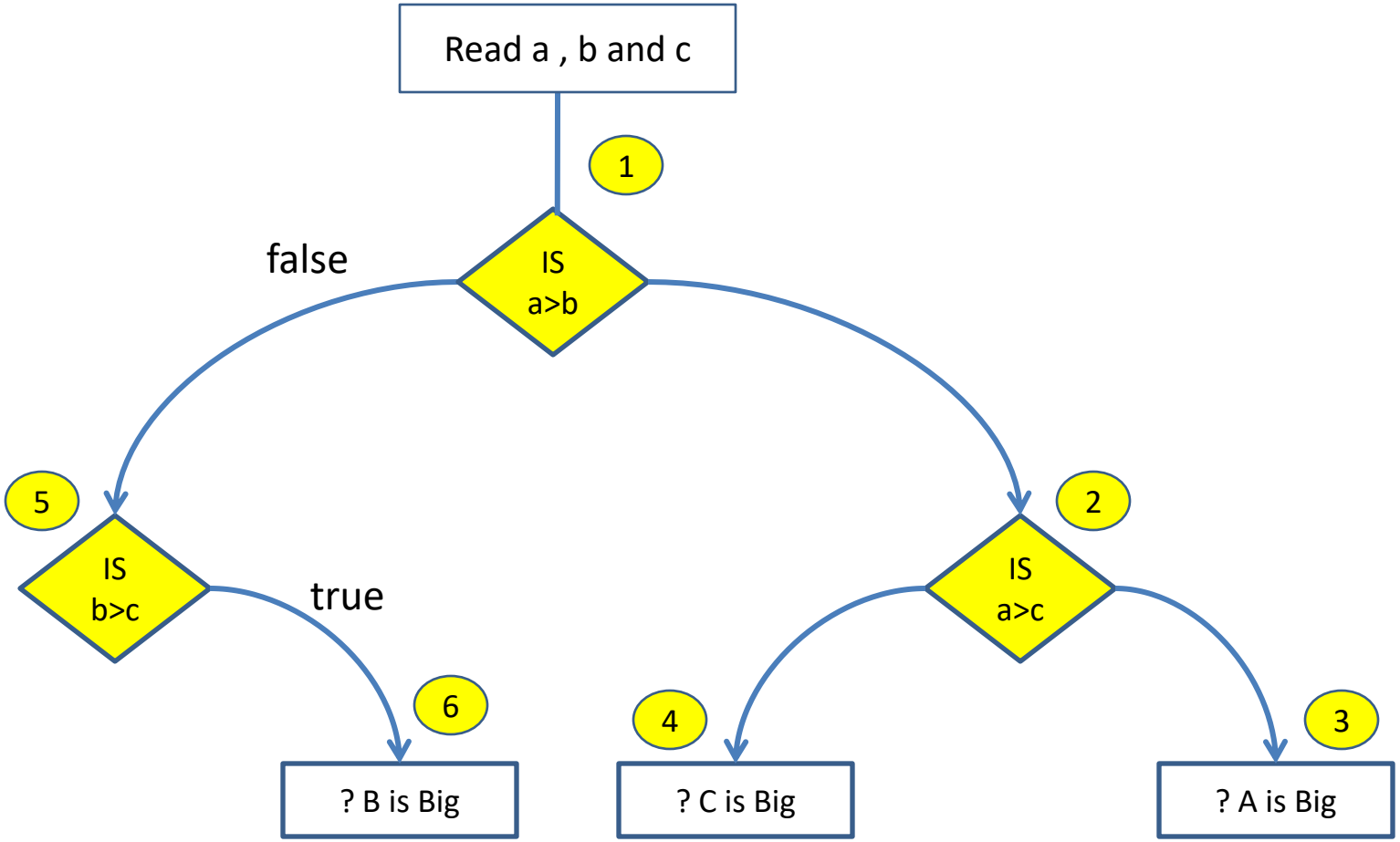

Read a , b and c
1
IS
a>b

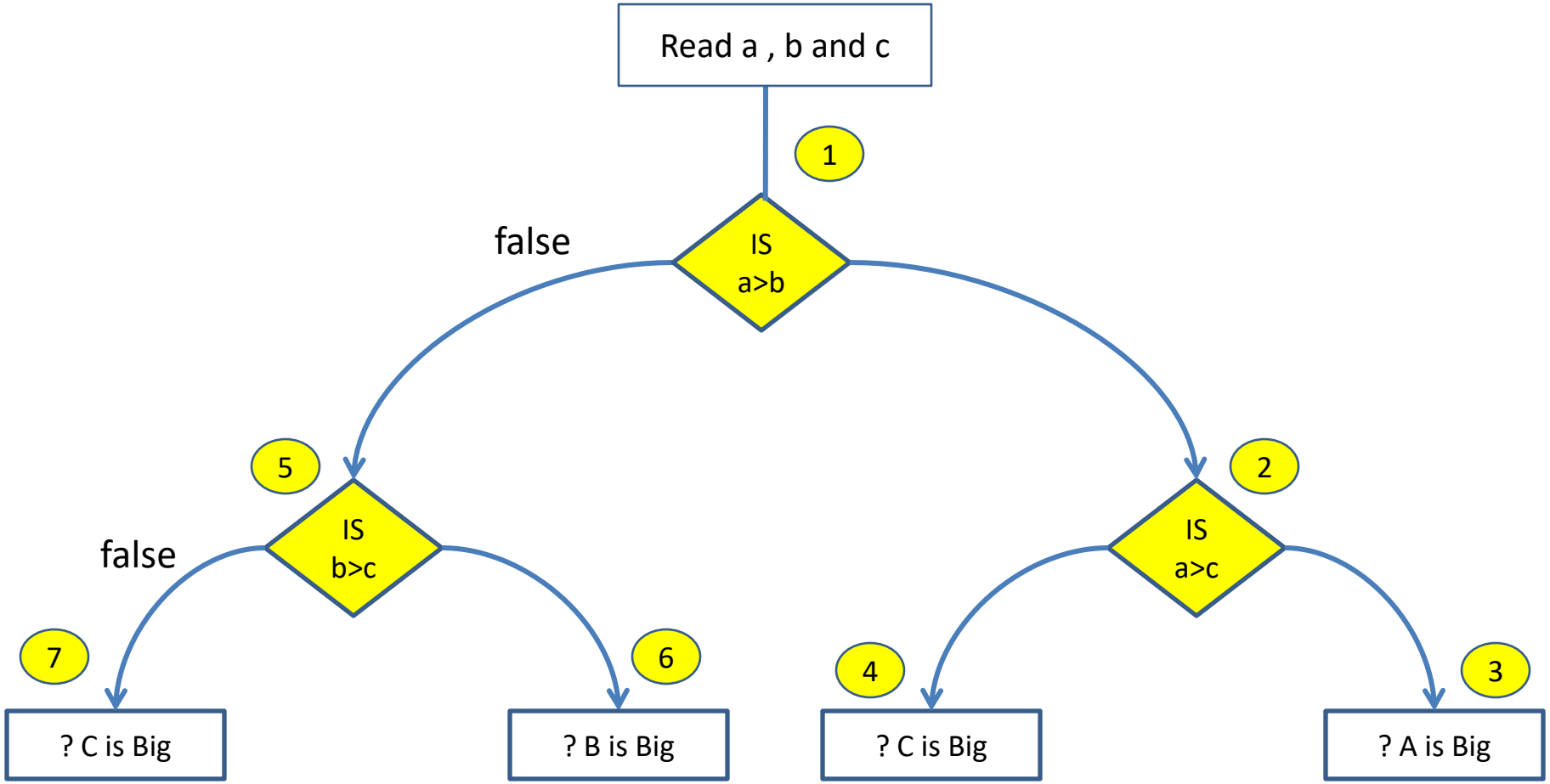

Read a , b and c
1
IS
a>b
true
2
IS
a>c

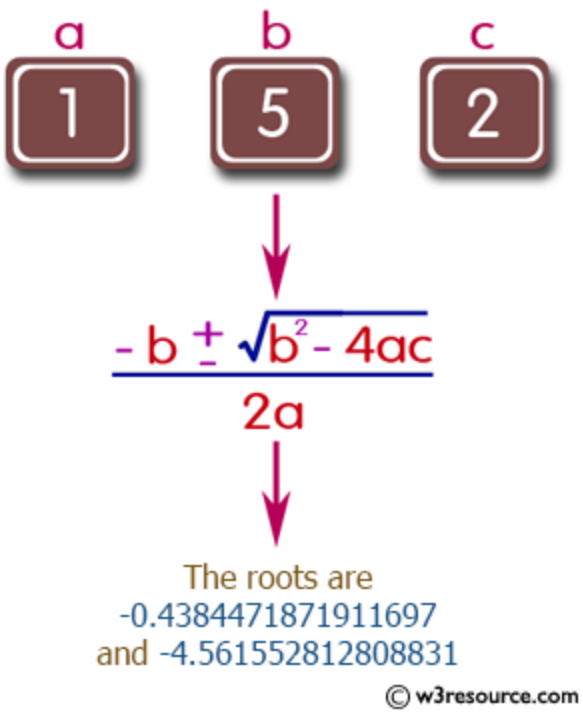

Read a , b and c
1
IS
a>b
true
2
IS
a>c
true
3
? A is Big


Read a , b and c
1
IS
a>b
true
2
IS
a>c
false
4
? C is Big
3
? A is Big


Read a , b and c
1
false
IS
a>b
5
IS
b>c
2
IS
a>c
4
? C is Big
3
? A is Big


Read a , b and c
1
false
IS
a>b
5
IS
b>c
true
2
IS
a>c
6
4
3
? B is Big
? C is Big
? A is Big


Read a , b and c
1
false
IS
a>b
5
2
false
IS
b>c
IS
a>c
7
6
4
3
? C is Big
? B is Big
? C is Big
? A is Big



Quadratic equation is $ax^2 + bx + c$

Determinant $D = b^2 - 4ac$

$D > 0$ means two real, distinct roots.

$D = 0$ means two real, identical roots/

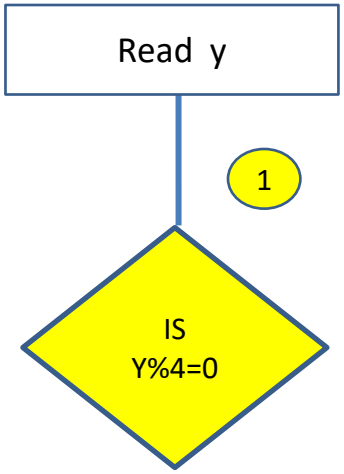
$D < 0$ means no real roots.

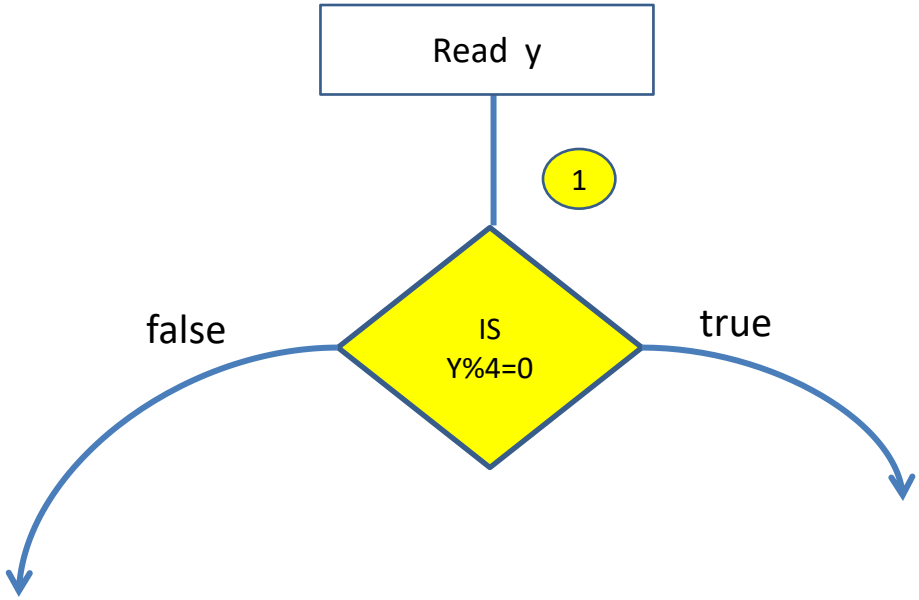
- Find out the given year Leap year or not

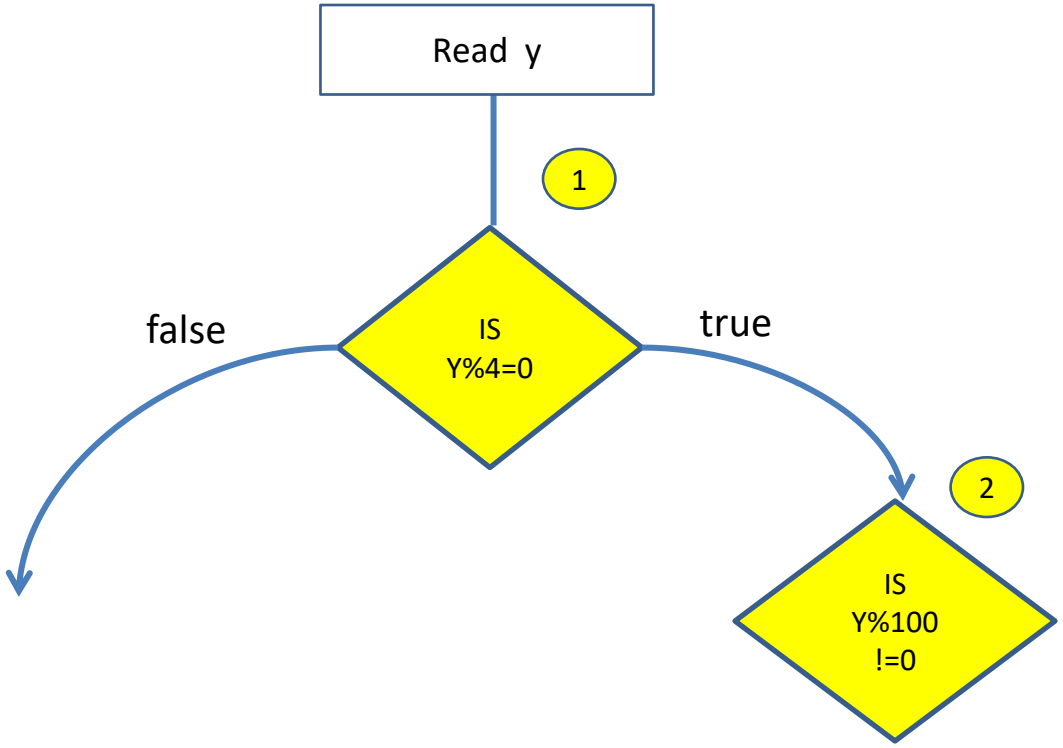
Leap year:

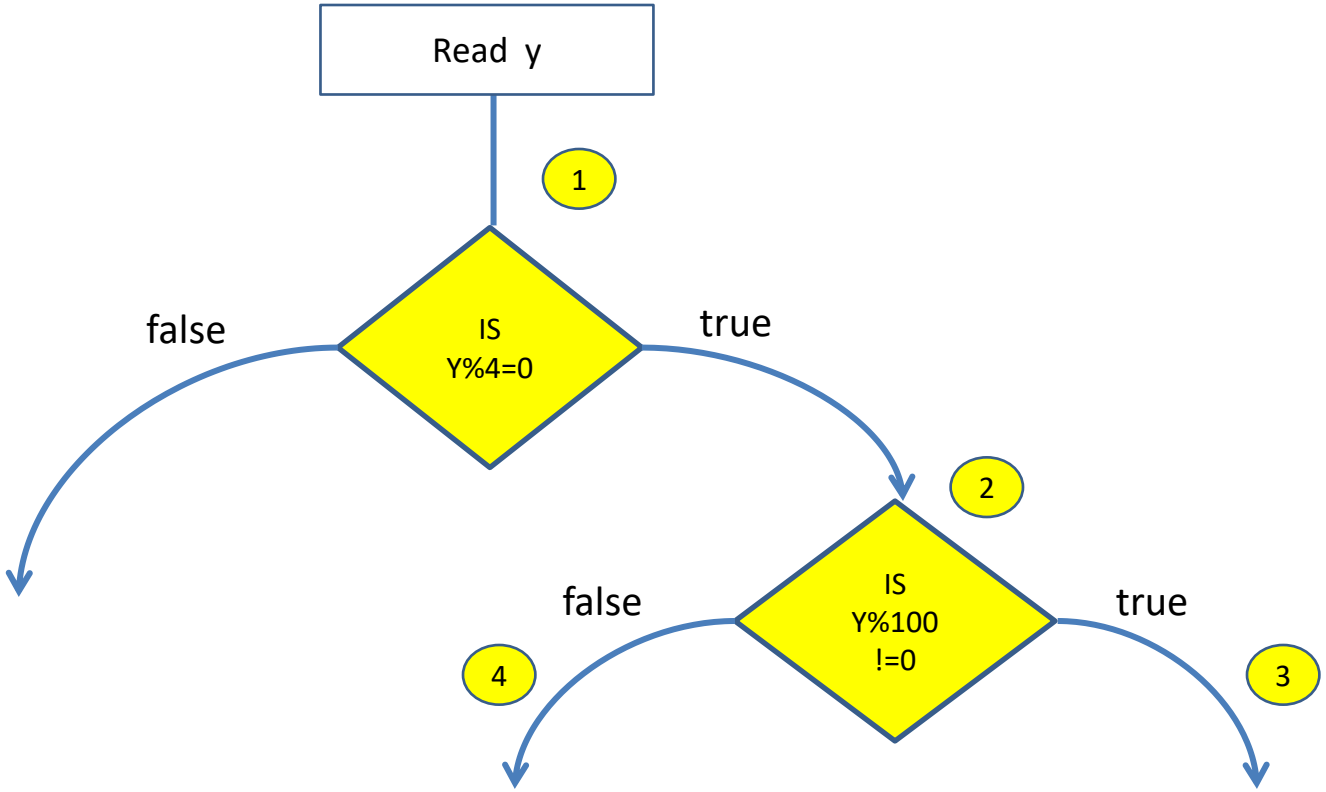
Once in 4 years (get divided by 4) and should not be a century. If it is a century then it should be 4th century (get divided by 400).

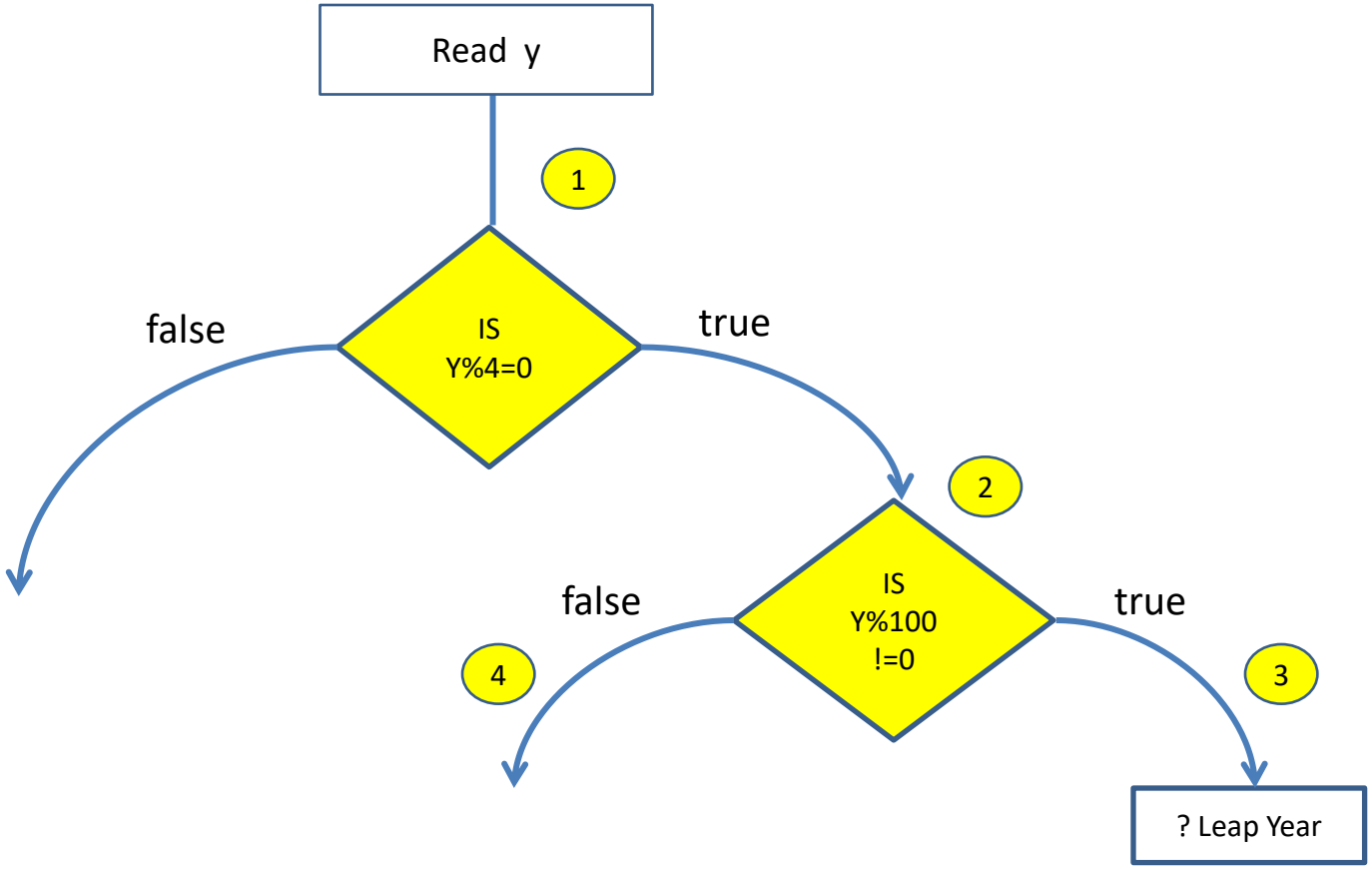
Read y

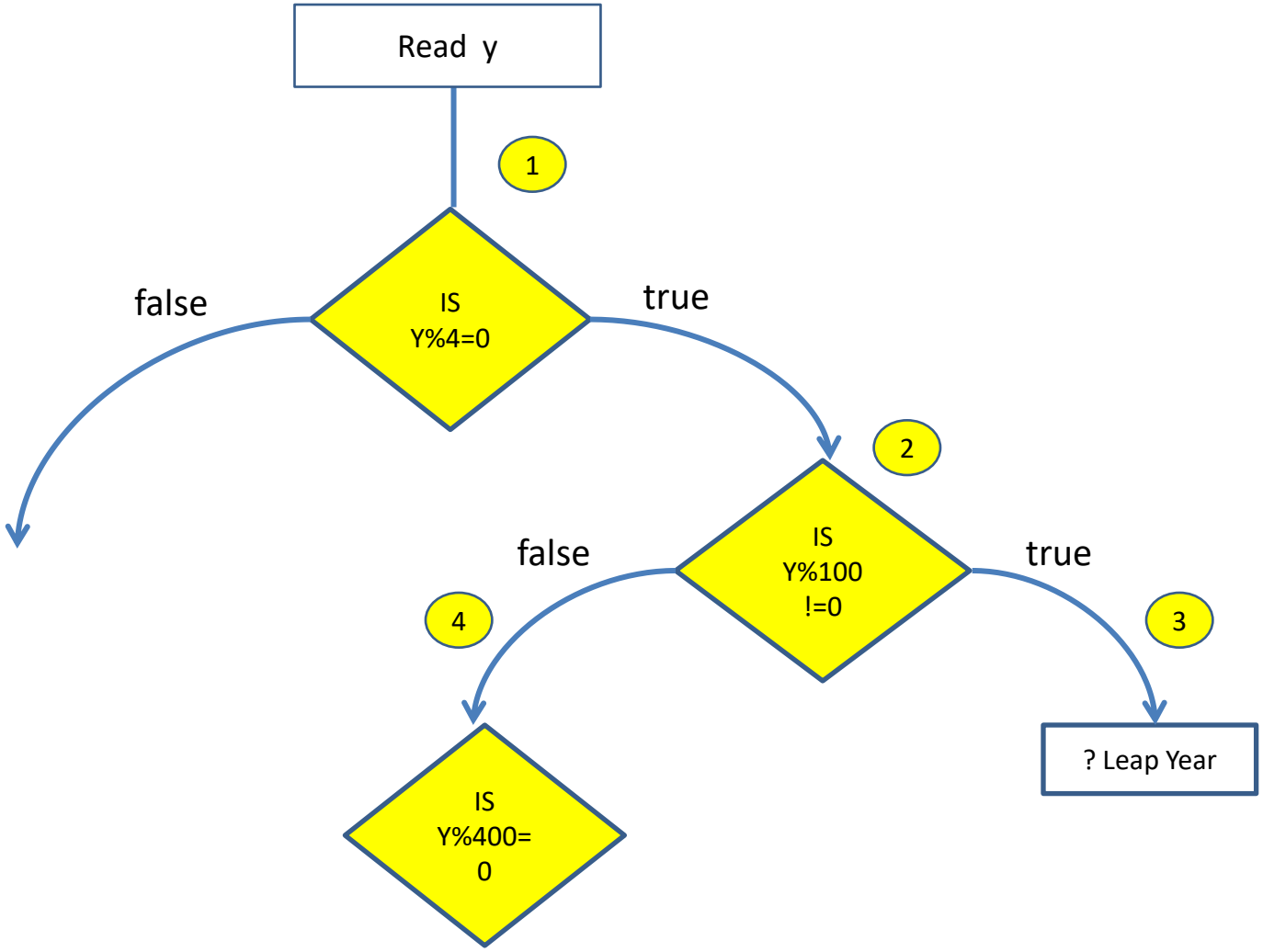

Read y
1
IS
Y%4=0

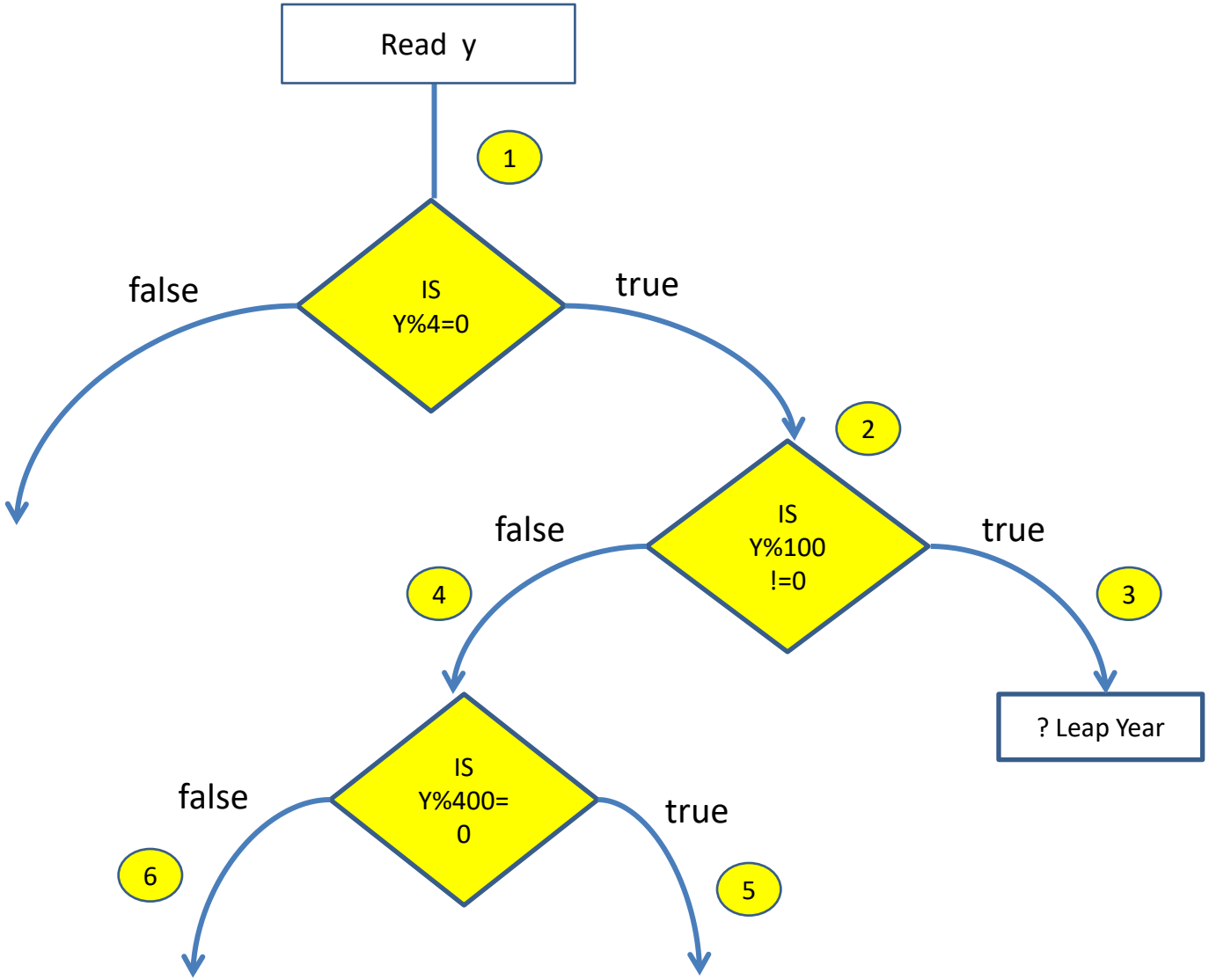

Read y
1
IS
Y%4=0
false
true

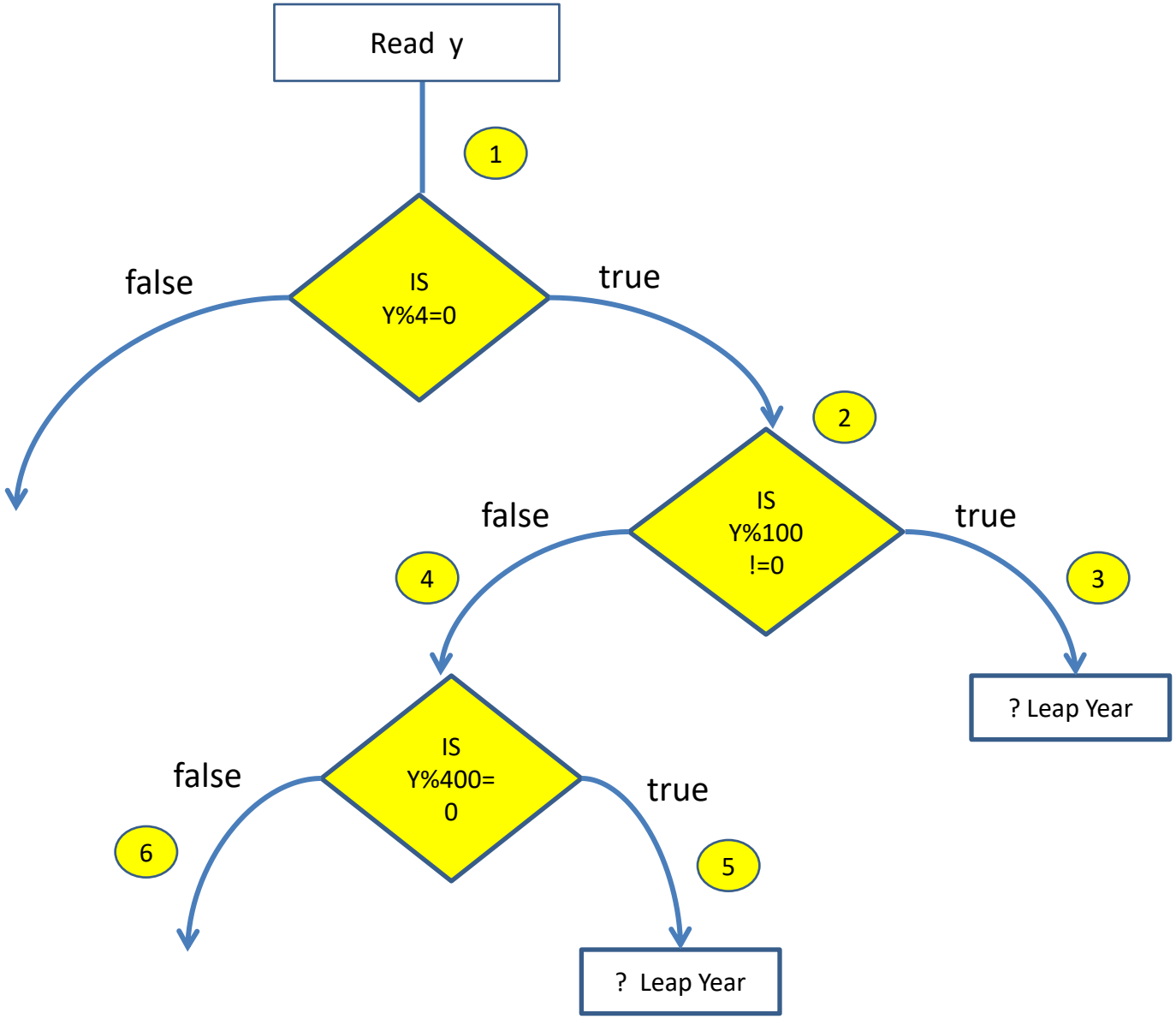

Read y
1
IS
Y%4=0
false
true
2
IS
Y%100
!=0

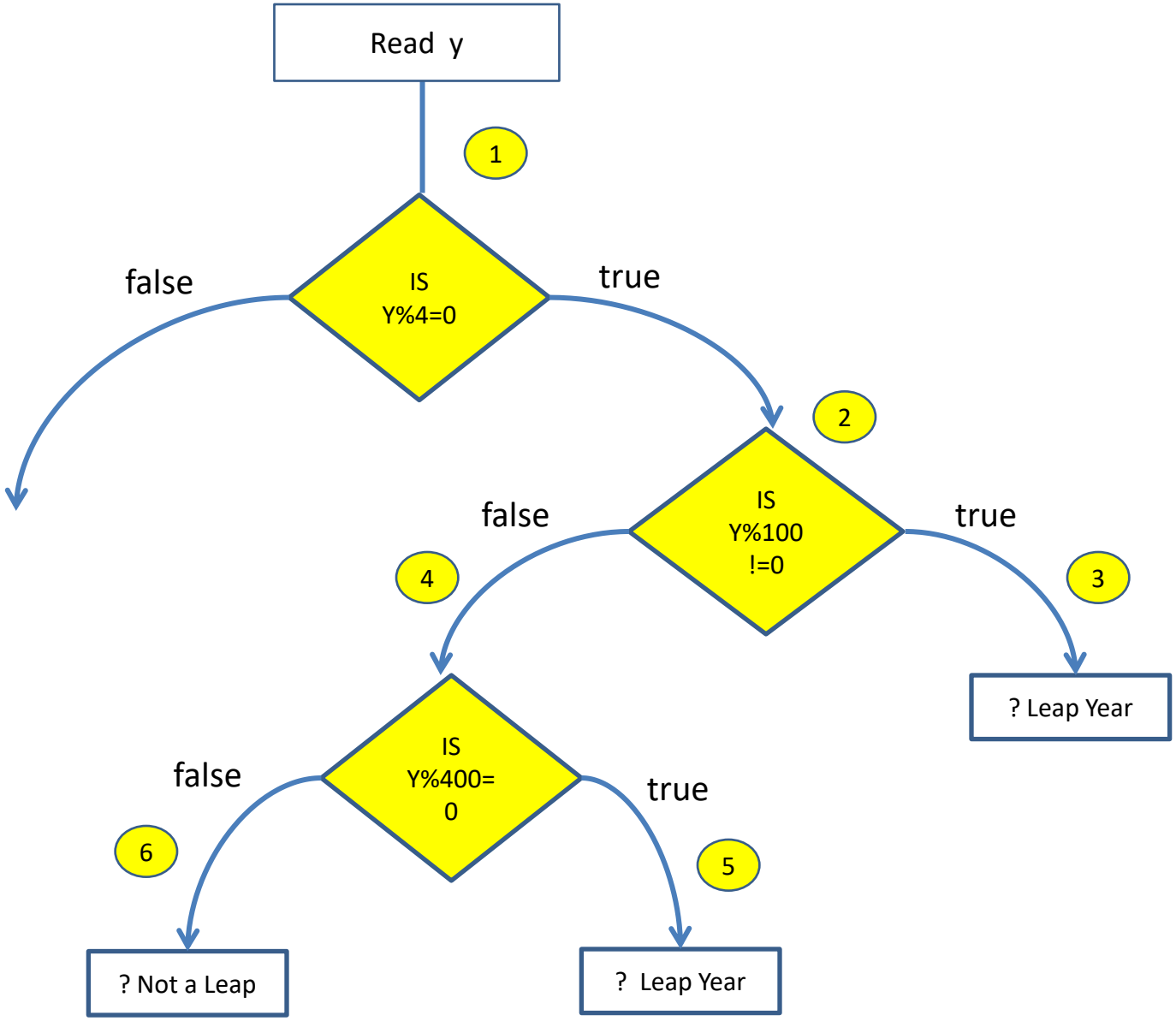

Read y
1
IS
Y%4=0
false
true
2
IS
Y%100
!=0
false
true
4
3

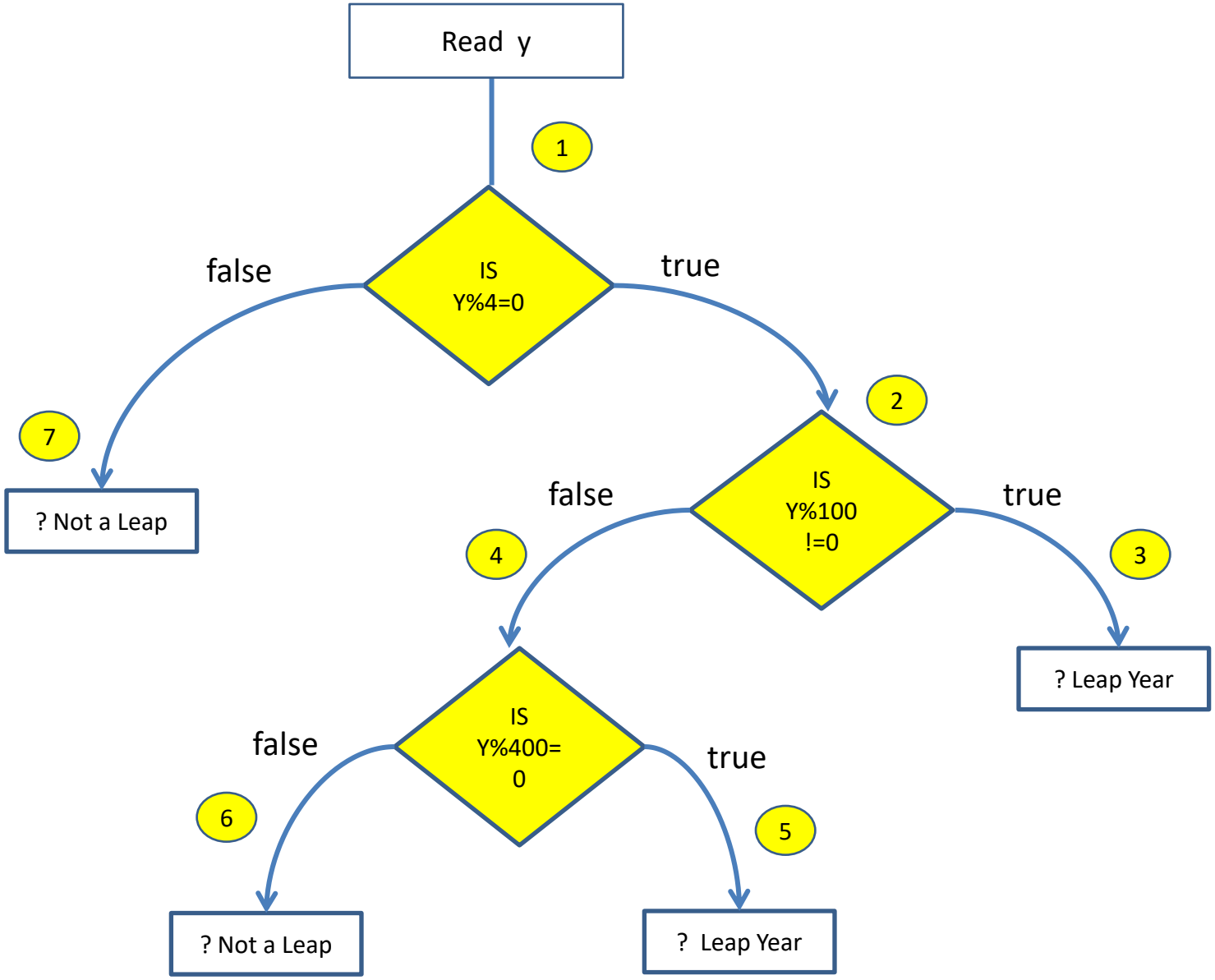

Read y
1
IS
Y%4=0
false
true
2
IS
Y%100
!=0
false
true
4
3
? Leap Year


Read y
1
IS
Y%4=0
false
true
2
IS
Y%100
!=0
false
true
4
3
IS
Y%400=
0
? Leap Year


Read y
1
IS
Y%4=0
false
true
2
IS
Y%100
!=0
false
true
4
3
? Leap Year
IS
Y%400=
0
false
true
6
5


Read y
1
IS
Y%4=0
false
true
2
IS
Y%100
!=0
false
true
4
3
? Leap Year
IS
Y%400=
0
false
true
6
5
? Leap Year


Read y
1
IS
Y%4=0
false
true
2
IS
Y%100
!=0
false
true
4
3
? Leap Year
IS
Y%400=
0
false
true
6
5
? Not a Leap
? Leap Year


Read y
1
IS
Y%4=0
false
true
7
? Not a Leap
2
IS
Y%100
!=0
false
true
4
3
? Leap Year
IS
Y%400=
0
false
true
6
5
? Not a Leap
? Leap Year

The Switch Statement

It is a multiple-branching statement where, based on condition, the control is transferred to one of the many possible points.

It provides the replacement for if statement, when:

1) There are series of possible decisions and
2) Decisions are based on CONSTANTS

The difference between if and switch statement is switch does not support relational and logical operators.

The Switch Statement

```
switch ( variable )   {
        case   value_1 :
              . . . . .
        break;
        case   value_2 :
              . . . . .
        break;
        . . .
        . . .
        case   value_n :
              . . . . .
        break;
        default :
              . . . . .
        break;
 }
```

Value can be only constant int or char.
It cannot be constant double
for ex:

```
case  1 :
case  2:

case  'y' :
case  'n' :

case  12.45 :    → its invalid
```

- Consider this scenario, take 2 numbers and a choice.

  If choice is 1 → print addition

  If choice is 2 → print subtraction

  If choice is 3 → print multiplication

  If choice is 4 → print division

  Other than 4 → print wrong choice

Now, you have 2 options if and switch

1. You can do the above program by using else if ladder.
2. You can do the above program by using switch case as it contains series of decisions and decision are based on constants.

- Consider this scenario, find the grade of the student based on avg marks

  > 70 → print A grade

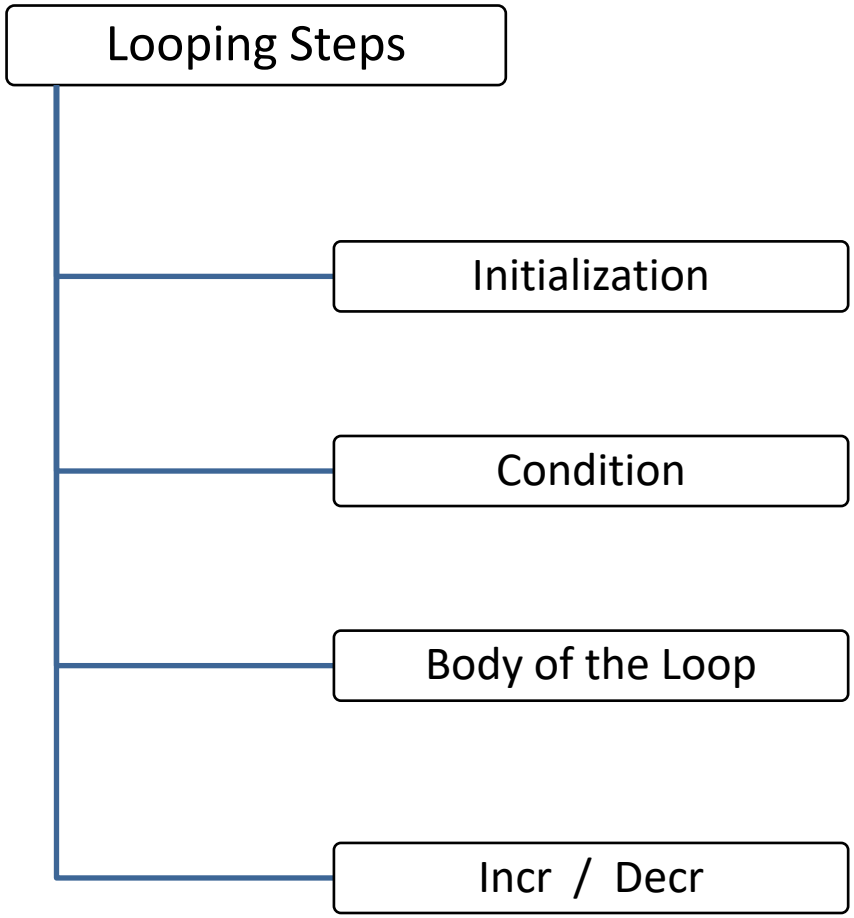
  50 – 70 → print B grade (avg >= 50 && avg < 70)

  35 – 50 → print C grade

  < 35 → print D grade

Now, here switch is not suitable for the above program as the decisions are not based on constants

Looping Style



Types of Loops

```
initialization ;

while ( condition )
{
        - - -
        - - - ( body )
        - - -
        incr  /  decr ;

}
```

while (entry control)

```
initialization ;

 do
 {
        - - -
        - - - ( body )
        - - -
        incr  /  decr ;

 } while ( condition ) ;
```

do.. while (exit control)

```
        1                2                4
for ( initialization  ;  condition  ;  incr / decr )
 {
        - - -     3
        - - - ( body )
        - - -
 }
```

for loop

Differences

The only difference between while and do while is do while executes the statements at least once irrespective of condition.

```
int n = 1;
while ( n < = 10 )  {
    System.out.println("JAVA");
    n++;
}
```

If n = 11 then **NO OUTPUT**

```
int n = 1;
do  {
    System.out.println("JAVA");
    n++;
} while( n < = 10 ) ;
```

If n = 11 then **1 TIME OUTPUT**

The difference between while and for is both are entry control, but while is variable entry control and for loop is fixed entry control

Enhancements of for loop:

1. Usage of comma operator (,) in for loop
2. Omit one of the steps
3. Creation of Dummy / null loops
4. for-each version

```
int i , j ;
for ( i=1 , j=10 ; i<=10 ; i++ , j-- )
{
        System.out.println ( i + "\t" + j ) ;
}
```

```
for ( ; i <= n ; )
```

```
for ( int i=1 ; i<=100000 ; i++ ) ;
```

→ it simply causes delay

Application using Loops

- Factorial
- Factors
- Prime or Not
- Fibonacci Series
- Sum of Each digit
- Reverse Order
- ArmStrong or Not (Sum of Cubes of each digit = Given Number eg: 153)
- Pallindrome or Not (Reverse of the Number = Given Number eg: 121)
- Strong or Not (Sum of factorials of each digit = Given Number eg: 145)
- Perfect or Not (Sum of factors except itself = Given Number eg: 6)

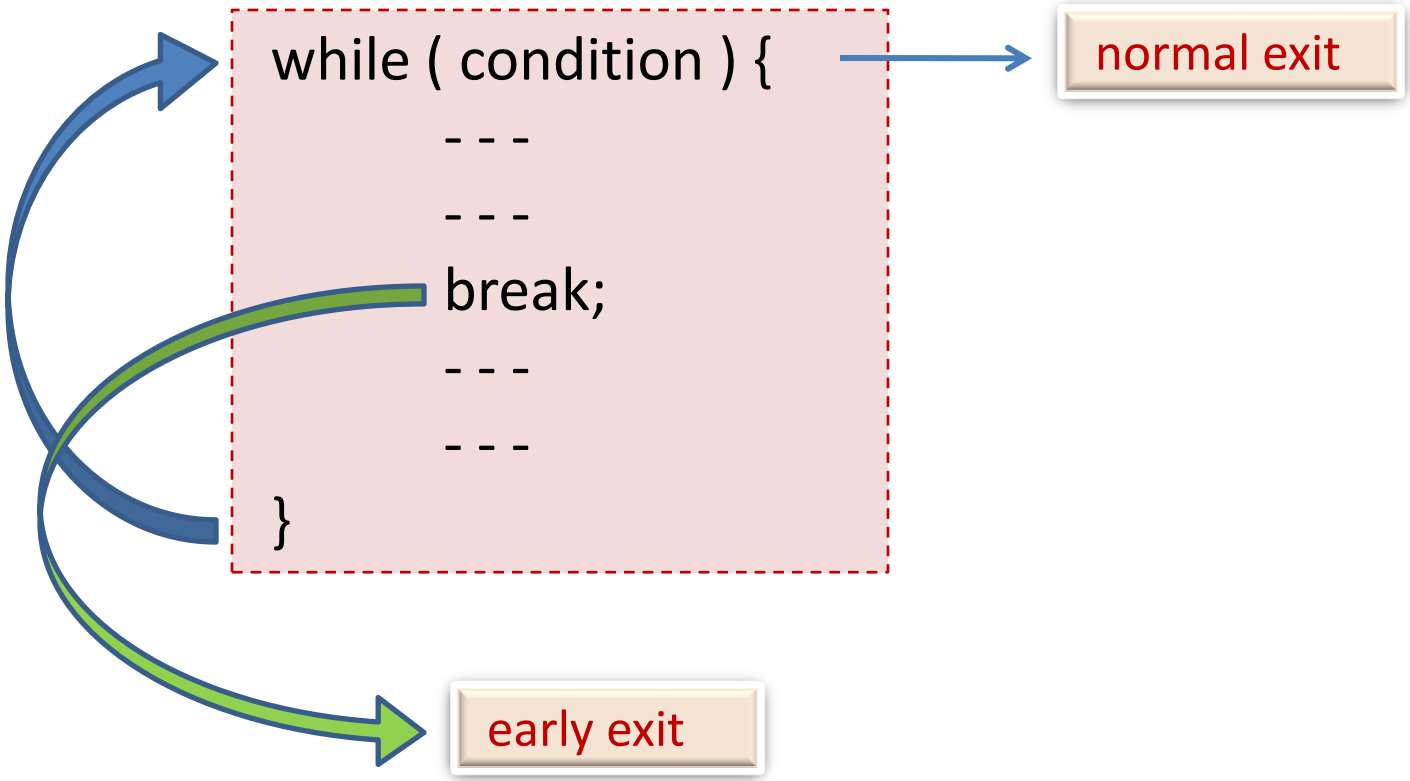
Branching Statements or Jump Statements

To improve the efficiency of the program

- break statement (early exit)
- continue (early iteration)
- return

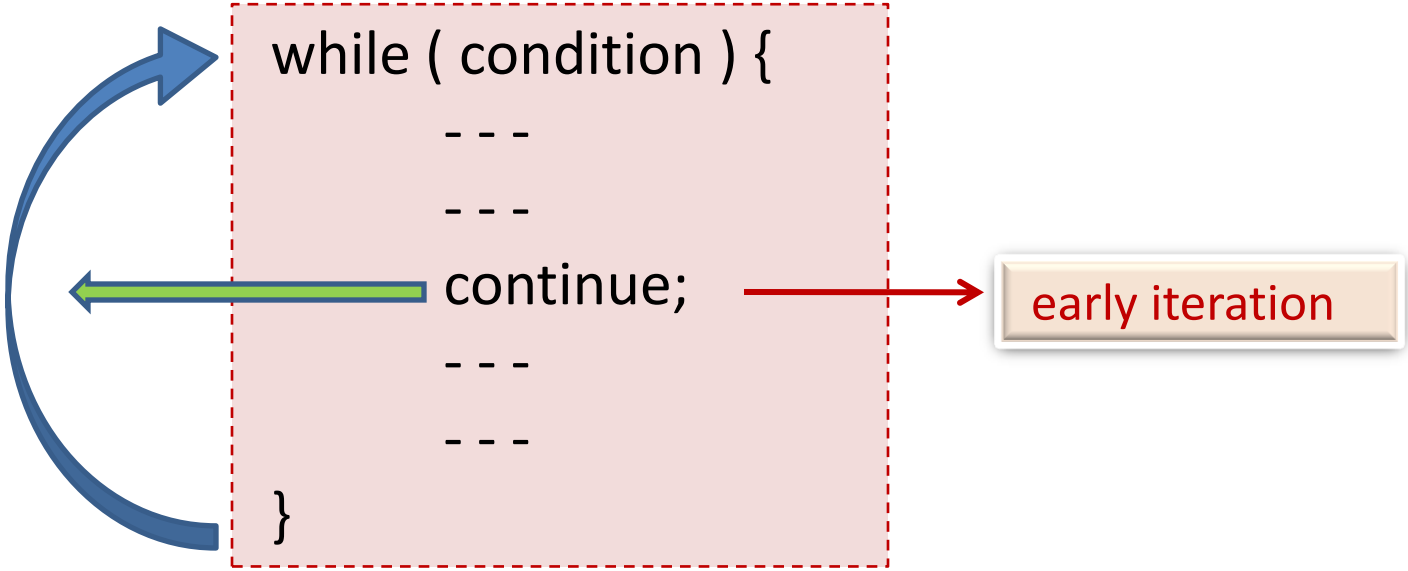
Break Statement

This statement breaks the currently repeating loop



Continue Statement

This statement forces the cursor to repeat from the next iteration by skipping subsequent statements



return Statement

```
class    Demo {
     public static void main ( String args [ ] ) {
            int   a = 5,  b = 10;
            System.out.println( a + b );
            return;
            System.out.println( a – b );
     }
}
```

Show compilation error saying unreachable code error

return Statement

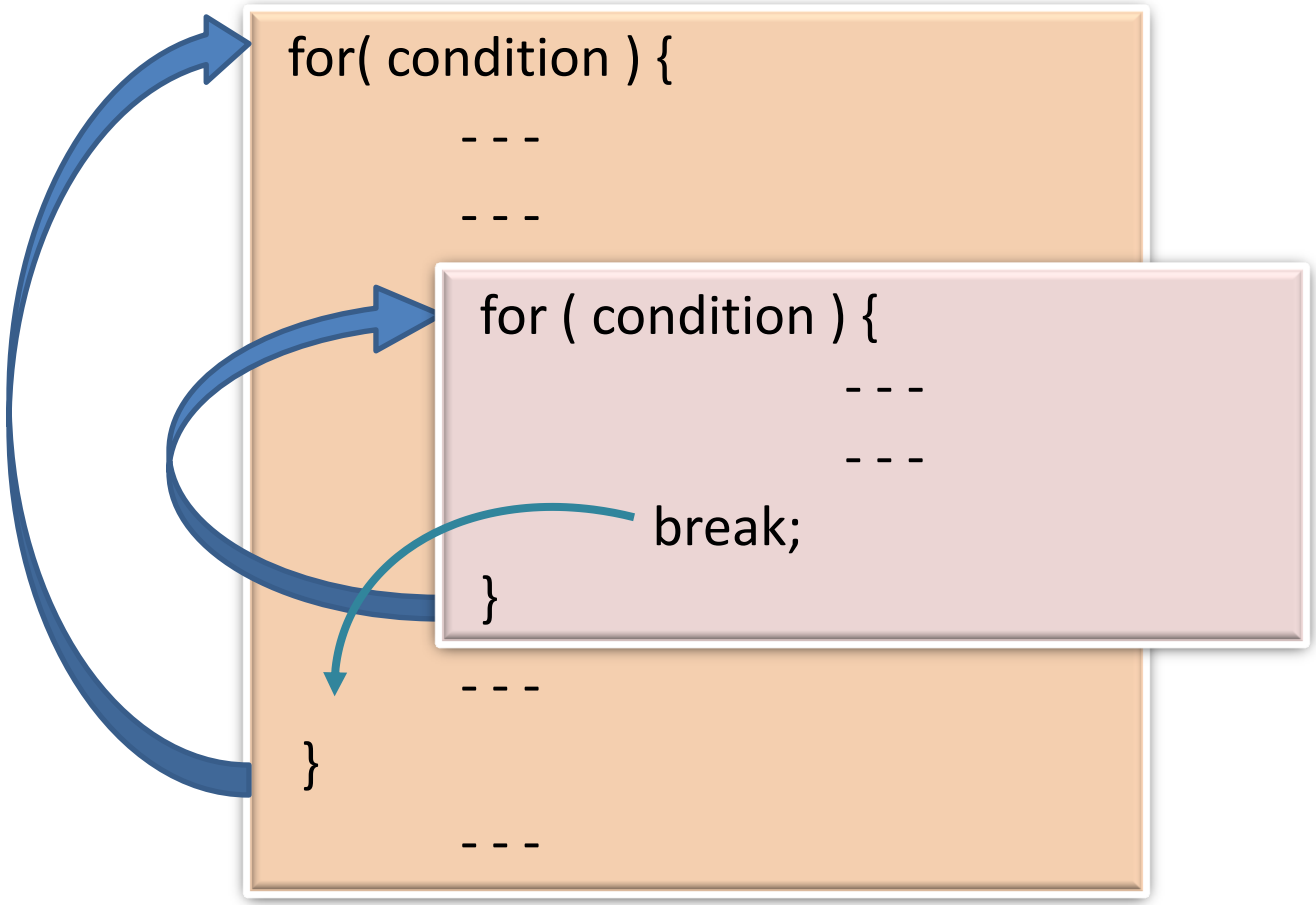
```
class    Demo {
    public static void main ( String args [ ] ) {
        int   a = 5,  b = 10;
        System.out.println( a + b );
        if ( a < b ) {
                return;  ⟶  System.exit ( 0 / 1 ) ;
        }
        System.out.println( a – b );
    }
}
```

1) Write a java program to find the real roots of a Quadratic equation.

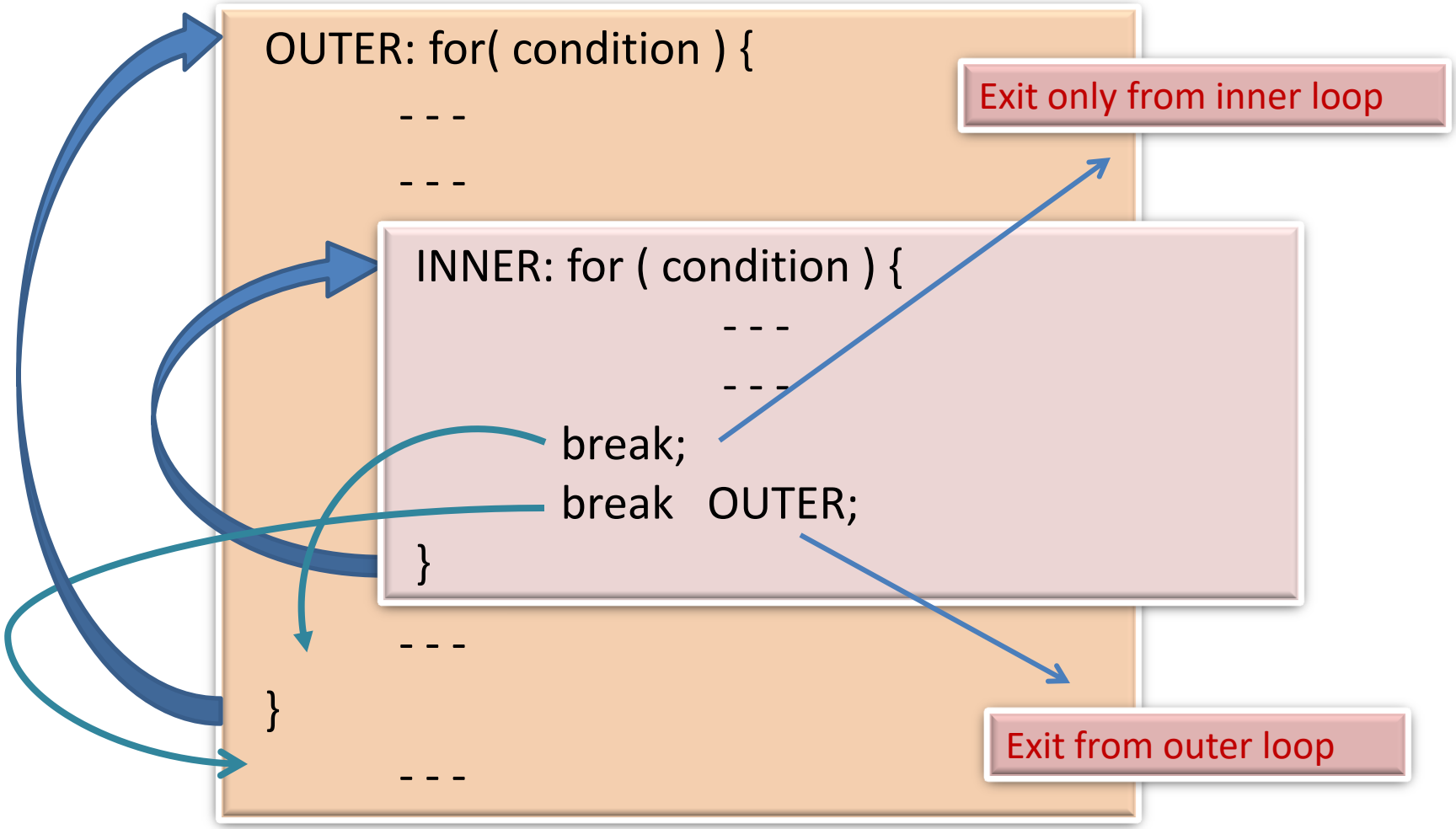
2)Write a java program to create menu with (control statements) and show the syntax when the user selects the option. Example(1. IF 2.IF-ELSE
3. Do-While
4. FOR
5. while
6. Switch
0. exit)

Iterate the menu till user selects 0.

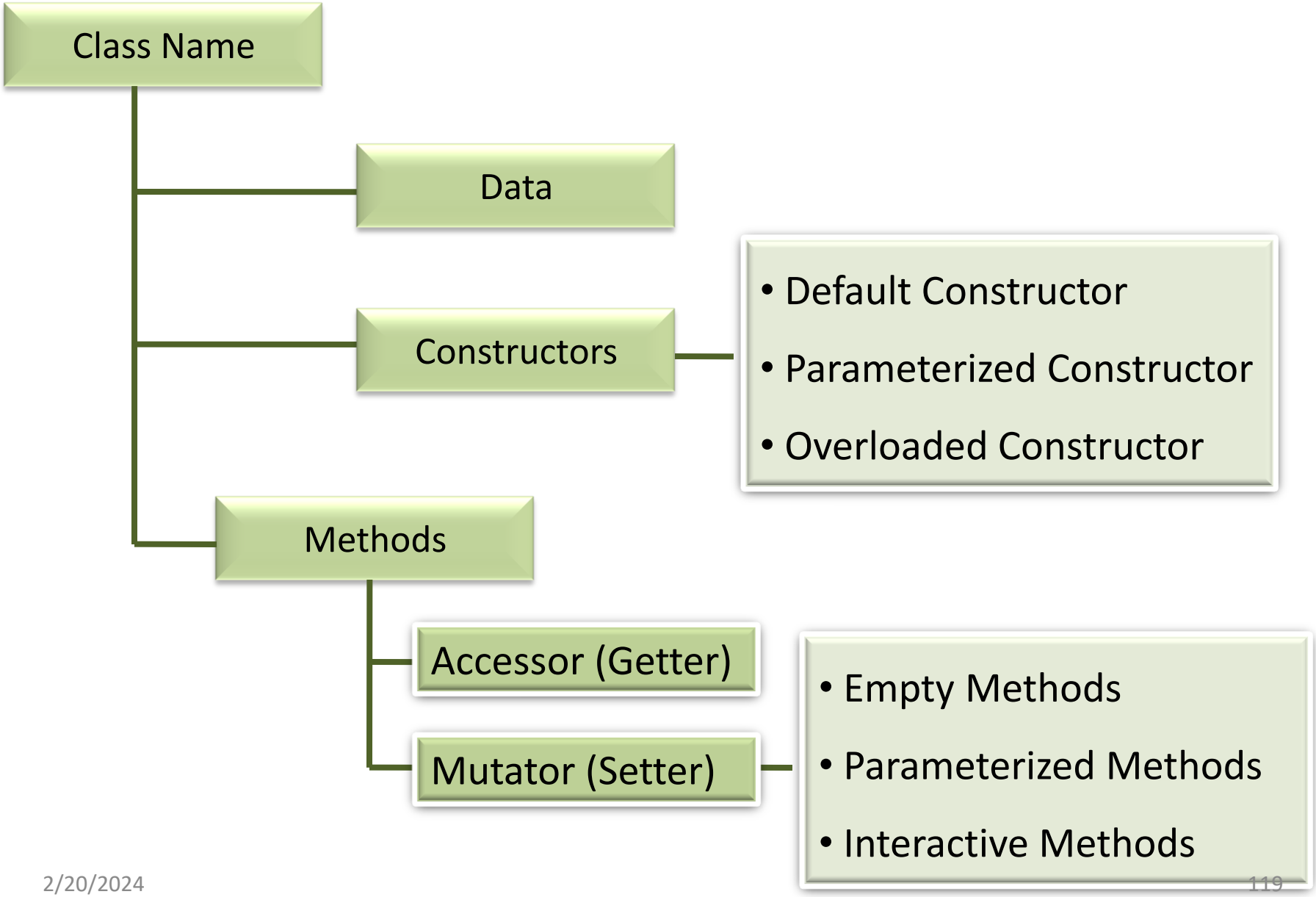
Labeled Statements



Labeled Statements



Components of a Class

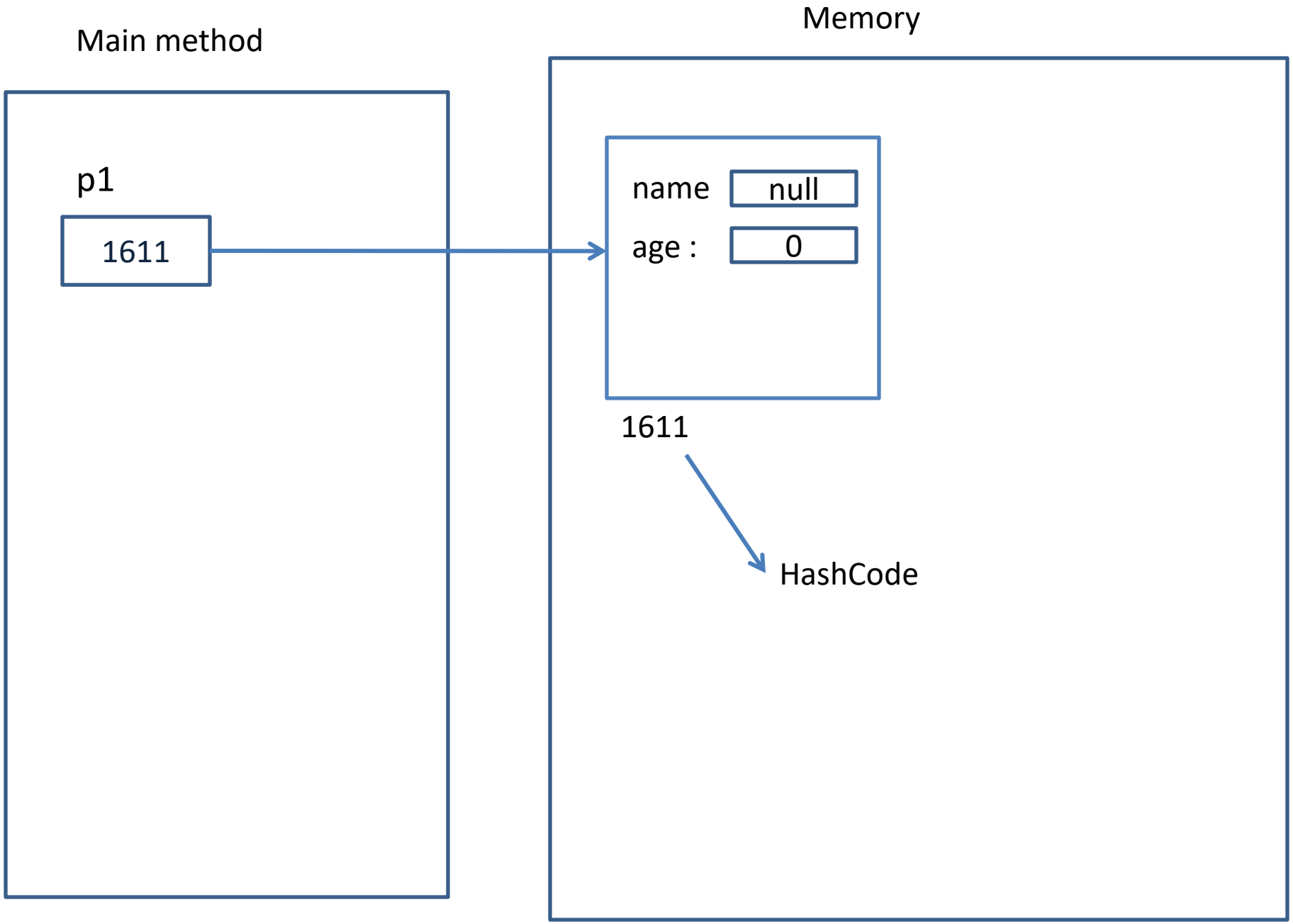


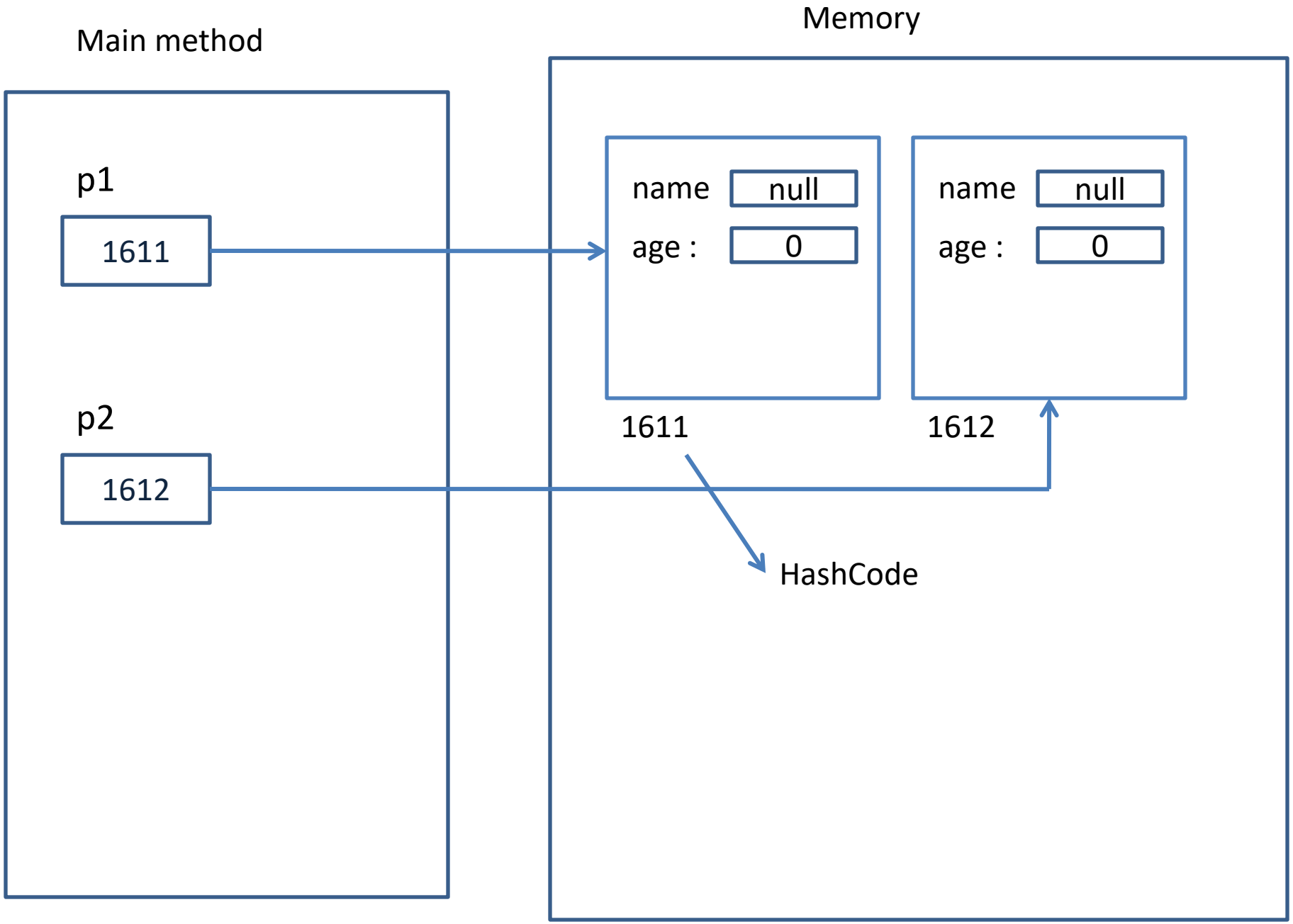
Main method

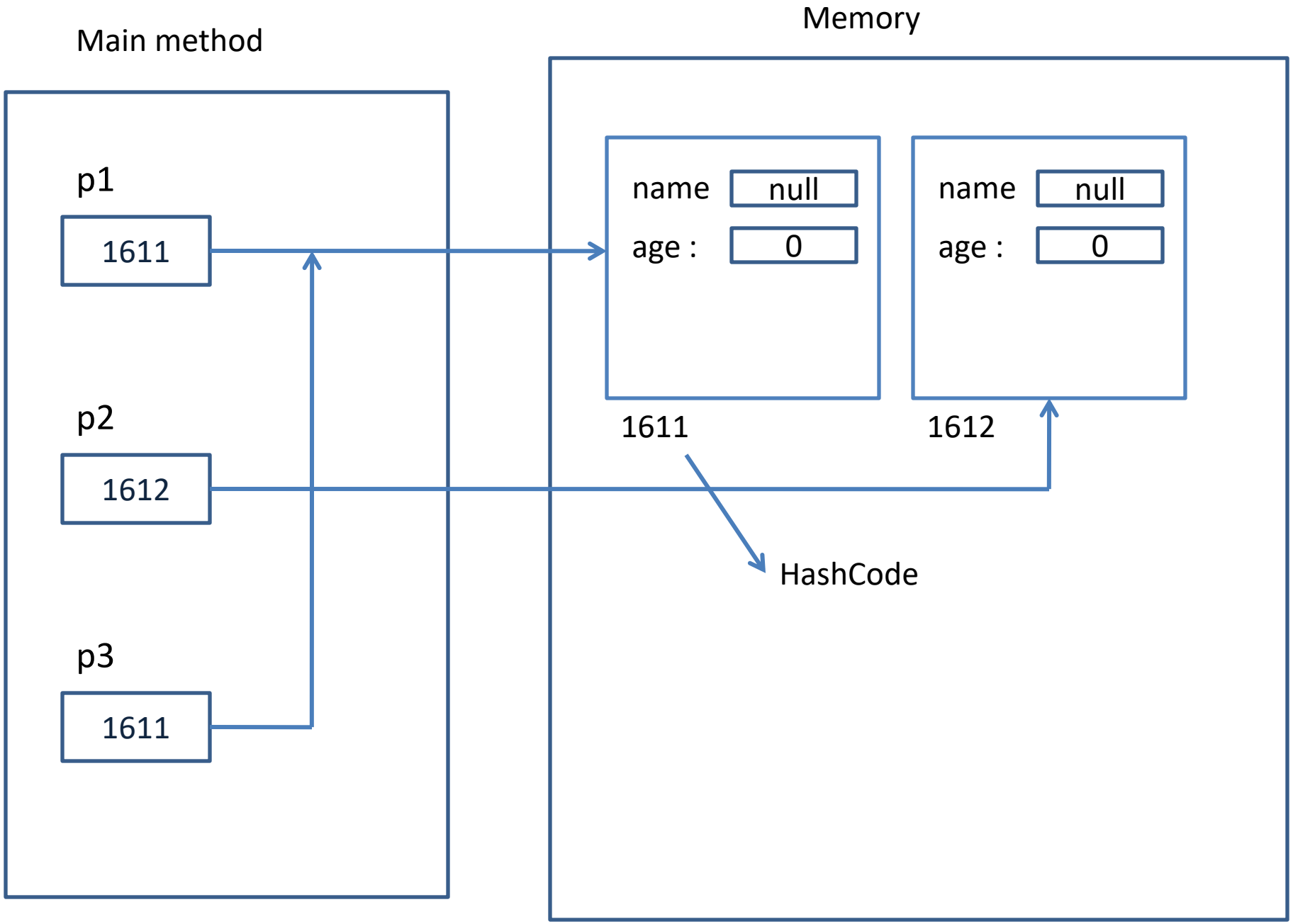
p1

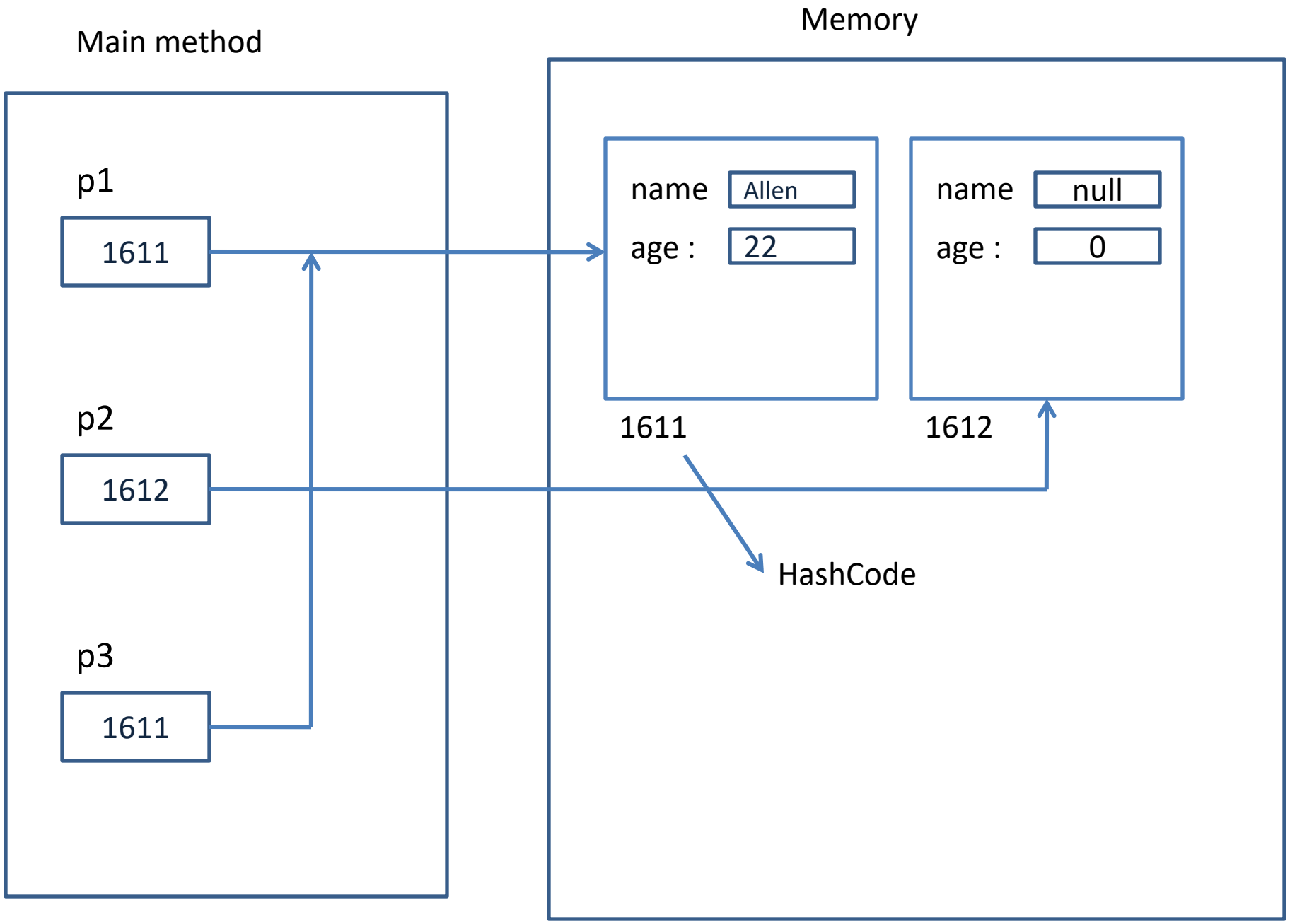
null

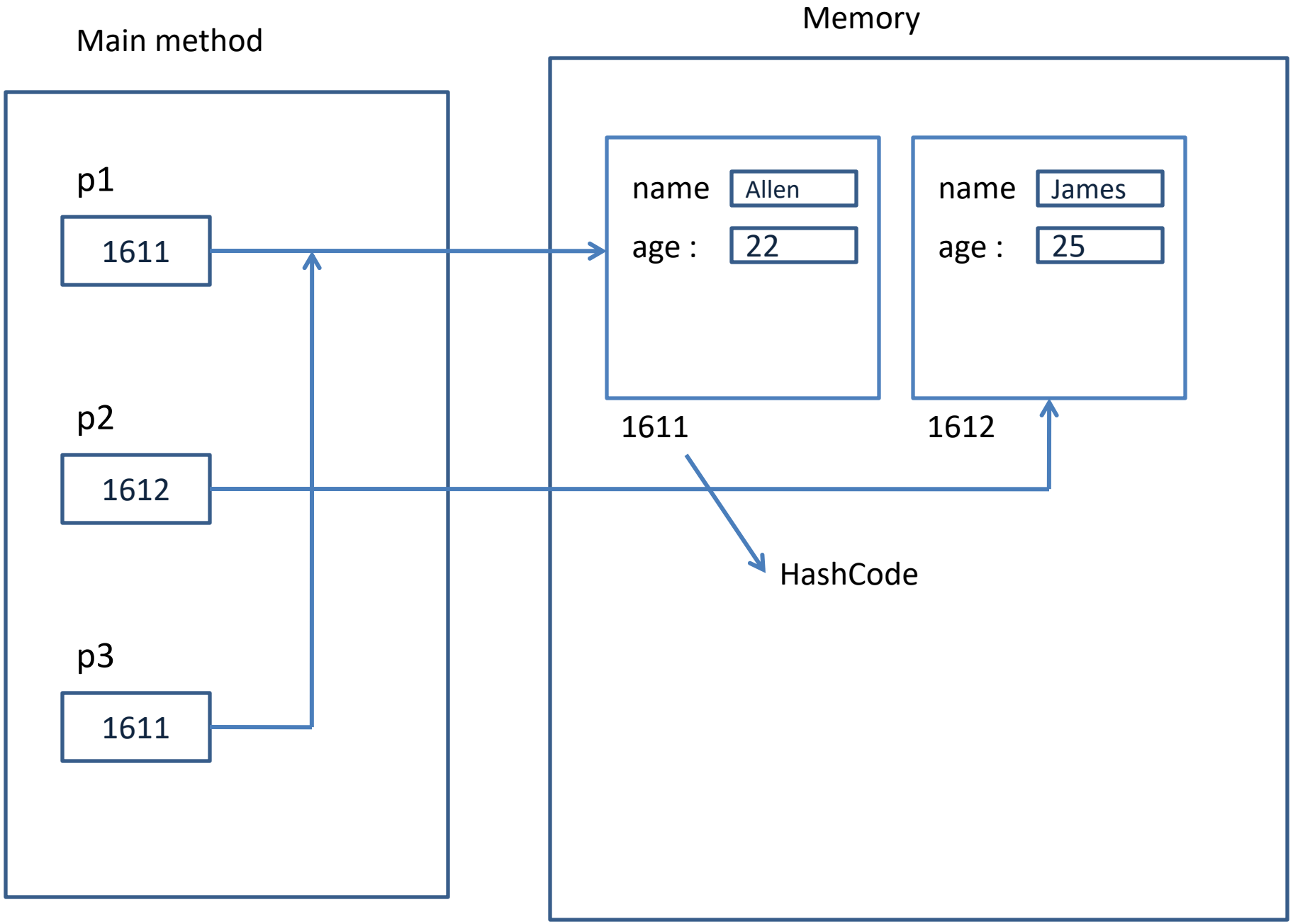
Memory

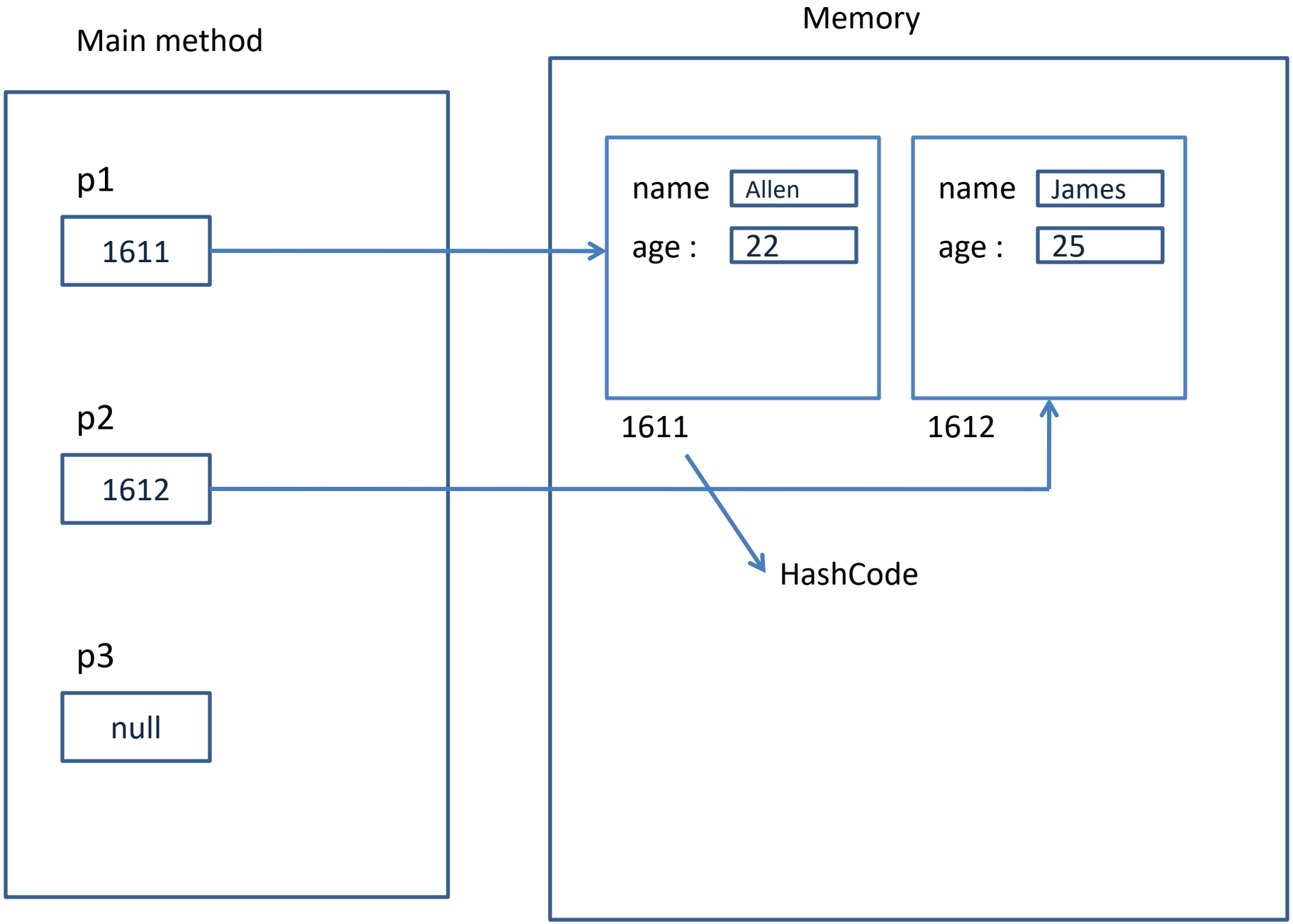

Main method
Memory
p1
1611
name
null
age :
0
1611
HashCode

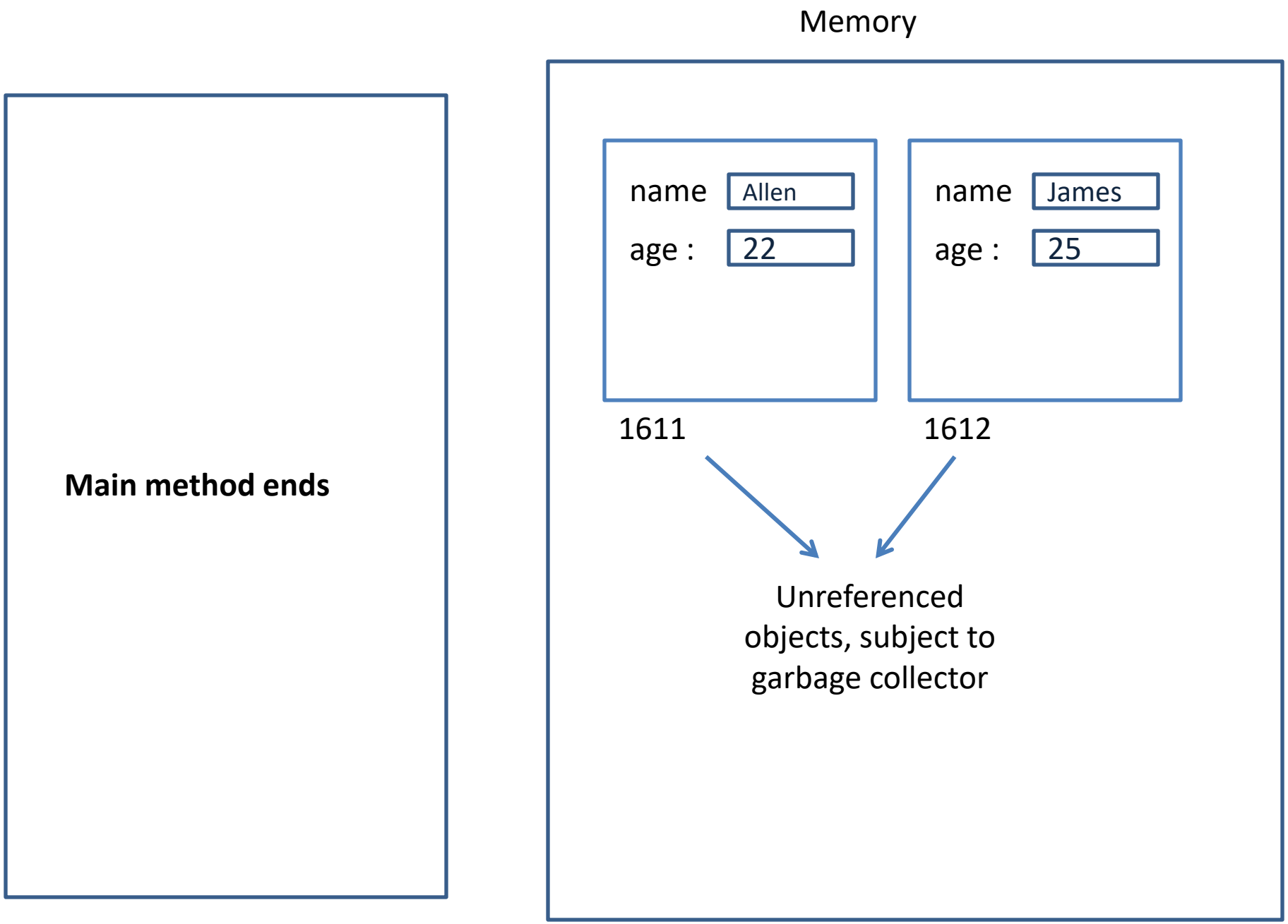

Main method
Memory
p1
1611
p2
1612
name
null
age :
0
1611
name
null
age :
0
1612
HashCode


Main method
Memory
p1
1611
p2
1612
p3
1611
name
null
age :
0
name
null
age :
0
1611
1612
HashCode


Main method
Memory
p1
1611
p2
1612
p3
1611
name
Allen
age :
22
name
null
age :
0
1611
1612
HashCode


Main method
Memory
p1
1611
p2
1612
p3
1611
name
Allen
age :
22
name
James
age :
25
1611
1612
HashCode


Main method
Memory
p1
1611
p2
1612
p3
null
name
Allen
age :
22
name
James
age :
25
1611
1612
HashCode


Memory
Main method ends
name
Allen
age :
22
1611
name
James
age :
25
1612
Unreferenced objects, subject to garbage collector