Java Basics

Everything we learned in Java

By: Rashedul Khan
Website: https://it.roti.io

Variable

Stores data

Holds information

Variables
[data_type] [name] = [data]

- Variables are named data in the sense that a variable stores data with a given name and type.

- [data] —> is a value such a whole number, fractional number, a letter, some words, or an object.

- [name] —> A descriptive name for the information that will be stored in this variable.

- [data_type] = The type of the data such as int, double, String, char, boolean.

Variable [type]

[type]	Data	Example
int	Whole Number	55
double	Decimal Number	55.6
char	Single Letter	‘A’
String	Sequence of letters	“Hello World”
boolean	true or false	TRUE

Variable Examples
[data_type] [name] = [data]

Statement	[data_type]	[name]	[data]
int a = 5;	int	a	5
double b = 5.7;	double	b	5.7
String c = “Hi”	String	c	“Hi”
boolean d = true	boolean	d	TRUE

Primitive Data Types

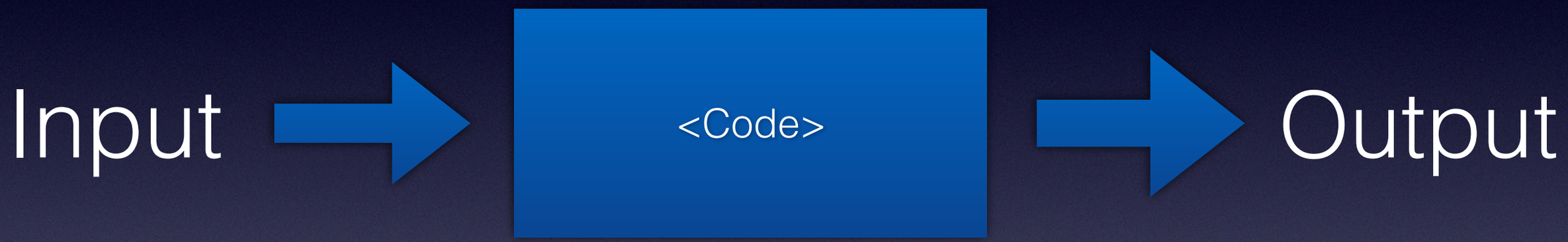
Types	Description	Size
boolean	True or false	1 bit
byte	Tiny number	8 bits
char	Letter	16 bits
short	Small number	16 bits
int	Number	32 bits
long	Long number	64 bits
float	Decimal point Number	32 bits
double	High precision decimal point number	64 bits

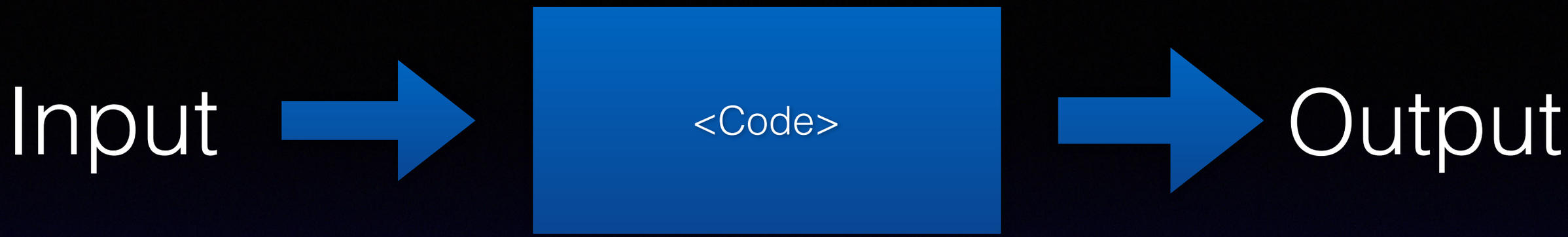
Method

Does something

Performs some action/job

Methods on a high-level






```
double calculateTax(double price)
{ <Code>}
```

Output Name Input

Method Call

```
add(5,7);
```

Output of add method is stored in a variable called result.

```
int result = add(5,7);
```

Methods
[output_type] [name]([input]) {<Code>}

- Methods are named blocks of code. It performs a job by taking some(or no) input and executing statements to produce an output(or no output).
- For example, a method that's supposed to add two numbers. Its job is to add two numbers that were passed in as arguments/input and produce an output that is the sum of those two inputs. The statement needed to accomplish this job is just the plus(+) operator

- int add(int a, int b) { return a + b; }

Method Format
[output_type] [name]([input]) {<Code>}

- [output_type] —> return data type such as int, double, String. Same as variable data types.

- [name] —> A descriptive name for the job/task that will be performed by the method

- [input] —> Data needed to perform the job/task. For example, in order to add two numbers the method needs to know those two numbers.

- <code> —> Sequence of statements to accomplish the job/task.

Method Examples
[output_type] [name]([input]) { <Code> }

	[output_type]	[name]	[input]	<Code>
Int add(int a, int b) { return a + b;}	int	add	int a, int b	return a +b;
void greet() { sout(“Hello!”); }	void = none	greet	none	sout(“Hello!”)
boolean isRich() { return true; }	boolean	isRich	none	return true;

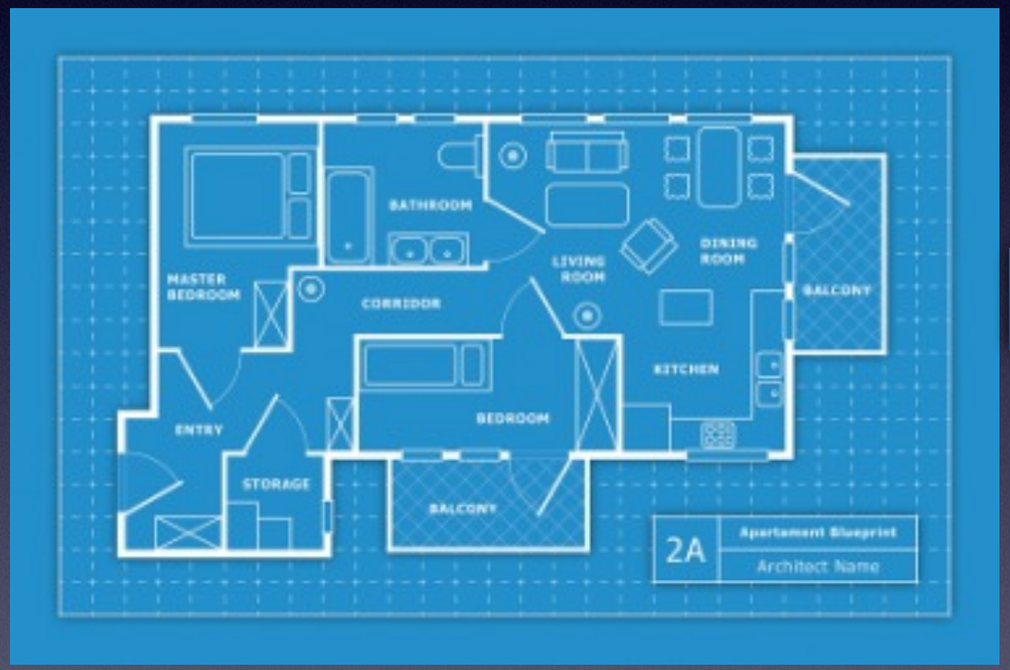
Class

Class

A blueprint for creating real objects.
In other words, what defines a object.

Just like real life

Class/Blueprint



Object/House

Blueprint

Object

Car.java

```
1  class Car {
2
3  }
4
```

Car.java Register.java Main.java

```
1  public class Main {
2
3      public static void main(String[] args)
4
5          Car car1 = new Car();
6
7      }
8
```

Class

Variable

Stores some data

Method

Performs a job

Objects

Are created from a class

Has things = Variables

Does things = Methods

Register.java | Car.java | GasTank.java

```java
public class Car {

    GasTank gasTank;

    public Car() {
        gasTank = new GasTank();
        gasTank.refill( gallons: 100);
    }

    public int drive() {
        gasTank.burnGas( gallons: 1);
        return gasTank.getGasLeftInTank();
    }

    public int getGasLeft() {
        return gasTank.getGasLeftInTank();
    }

}
```

Static

Associated to a class,
not to an instance

Register.java | Car.java | GasTank.java

```java
public class GasTank {

    public static int capacityInGallons = 10;

    int gasLeftInTank = 0;

    public void refill(int gallons) {
        gasLeftInTank = gasLeftInTank + gallons;
    }

    public int burnGas(int gallons) {
        int result = gasLeftInTank - gallons;
        gasLeftInTank = result;
        return result;

    }

    public int getGasLeftInTank() {
        return gasLeftInTank;
    }

}

```

Access static

Register.java | Main.java | Car.java | GasTank.java

```
public class Main {

    public static void main(String[] args) {
        int capacity = GasTank.capacityInGallons;
        System.out.println(capacity);
    }

}
```

Create the following class

BankAccount Blueprint

- Has balance
- Can take Deposits
- Can Show balance
- Can Withdraw
- Associated to a person

Conditionals

Conditionals

Decision making logic.

If this is true then do this…

Conditionals

- If ([condition]) { // Execute this block }
- Else if ([condition]) { // Execute this block }
- Else if ([condition]) { // Execute this block }
- Else { // Execute this block }

[condition]

- The condition must be boolean
- Operators that returns boolean are listed on the next slide

[condition] - Operators

Condition	TRUE if…
If (a == b)	A equals to b
If (a != b)	A is NOT equal to b
If (a < b)	A is less than b
If (a > b)	A is greater than b
If (a <= b)	A is less than or equal to b
If (a >= b)	A is greater than or equal to b

Arrays

Arrays

Stores more than one data

Holds multiple information

However…

- Fixed size. Meaning the size of the array cannot be changed once created.
- Must know size of array before creating one.
- All elements must have the same type. Meaning an array of integers can store only integers. Not any other types such as String or double.
- If you want to store integers and Strings you must create two separate arrays.

Creating an Array

- 2 ways to create an Array
  - Array with values
  - Empty array

Creating Array with values

- int[] numbers = { 1, 2, 3, 4, 5 };
- double[] fractions = { 1.2, 2.3, 3.5, 4.8, 5.5 };
- String[] menu = { “Tea”, “Coffee”, “Cookies” };

Creating Empty Array

- int[] numbers = new int[5];
- double[] fractions = new double[5];
- String[] menu = new String[3];

Accessing Array Elements

String[] menu = { “Tea”, “Coffee”, “Cookies” };

- 0th element —> menu[0] —> “Tea”
- 1st element —> menu[1] —> “Coffee”
- 2nd element —> menu[2] —> “Cookies”
- 3rd element —> menu[3] —> Error

Putting data in Arrays

String[] menu = { “Tea”, “Coffee”, “Cookies” };

- Overwrite 0th element —> menu[0] = “Chai”
- Overwrite 1st element —> menu[1] = “Thai Coffee”
- Overwrite 2nd element —> menu[2] = “Biscuit”
- Overwrite 3rd element —> menu[3] = “Cake” —> Error

Loops

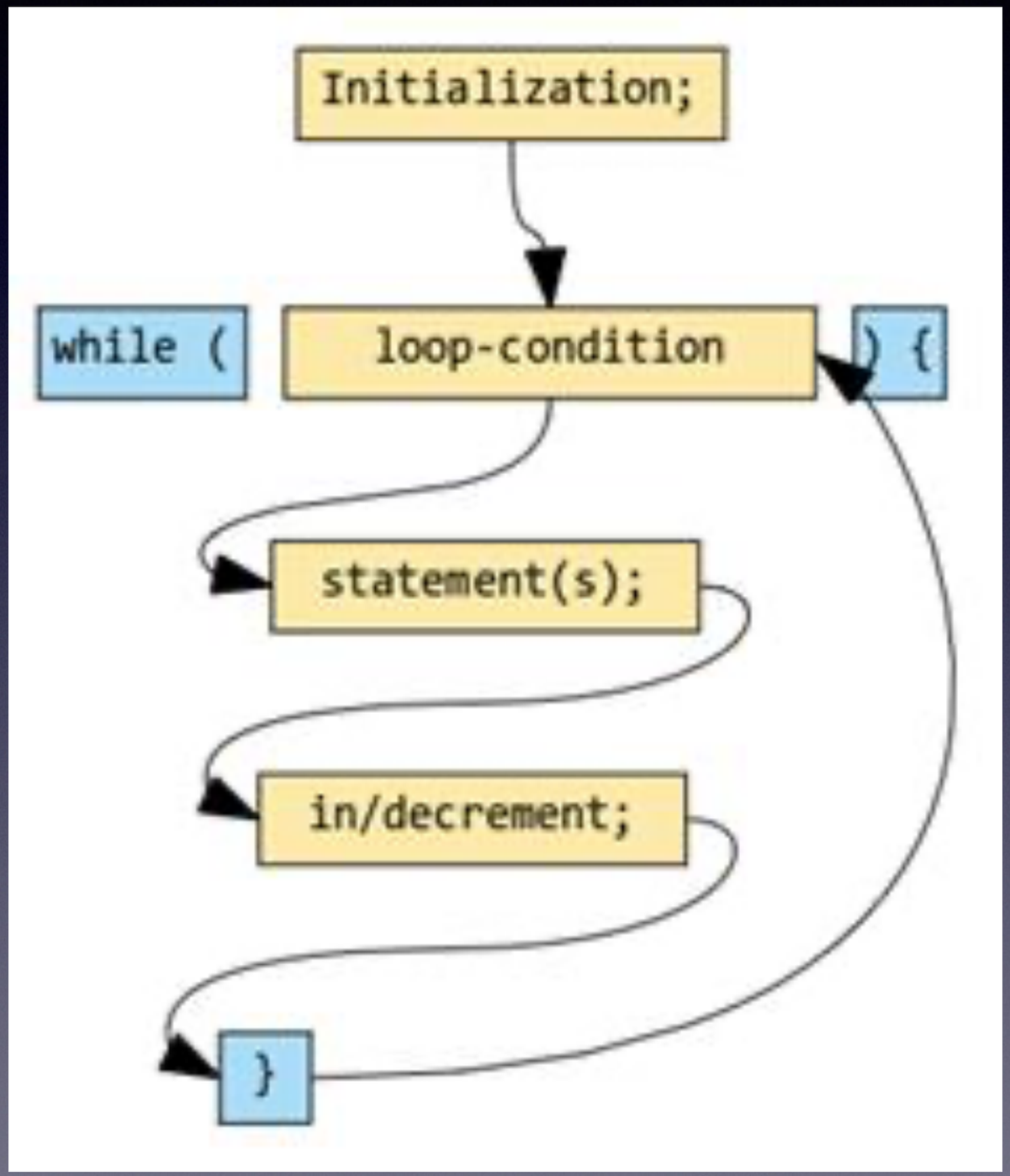
Loops

- Performs a task numerous times.
- Repeats a block of code until the condition becomes false or explicitly break out of loop.
- In other words, it will repeat as long as condition is true.

Types of Loop

- While loop
- For loop
- For Each

While Loop


Initialization;
while (
loop-condition
) {
statement(s);
in/decrement;
}

While Loop - Never Ending

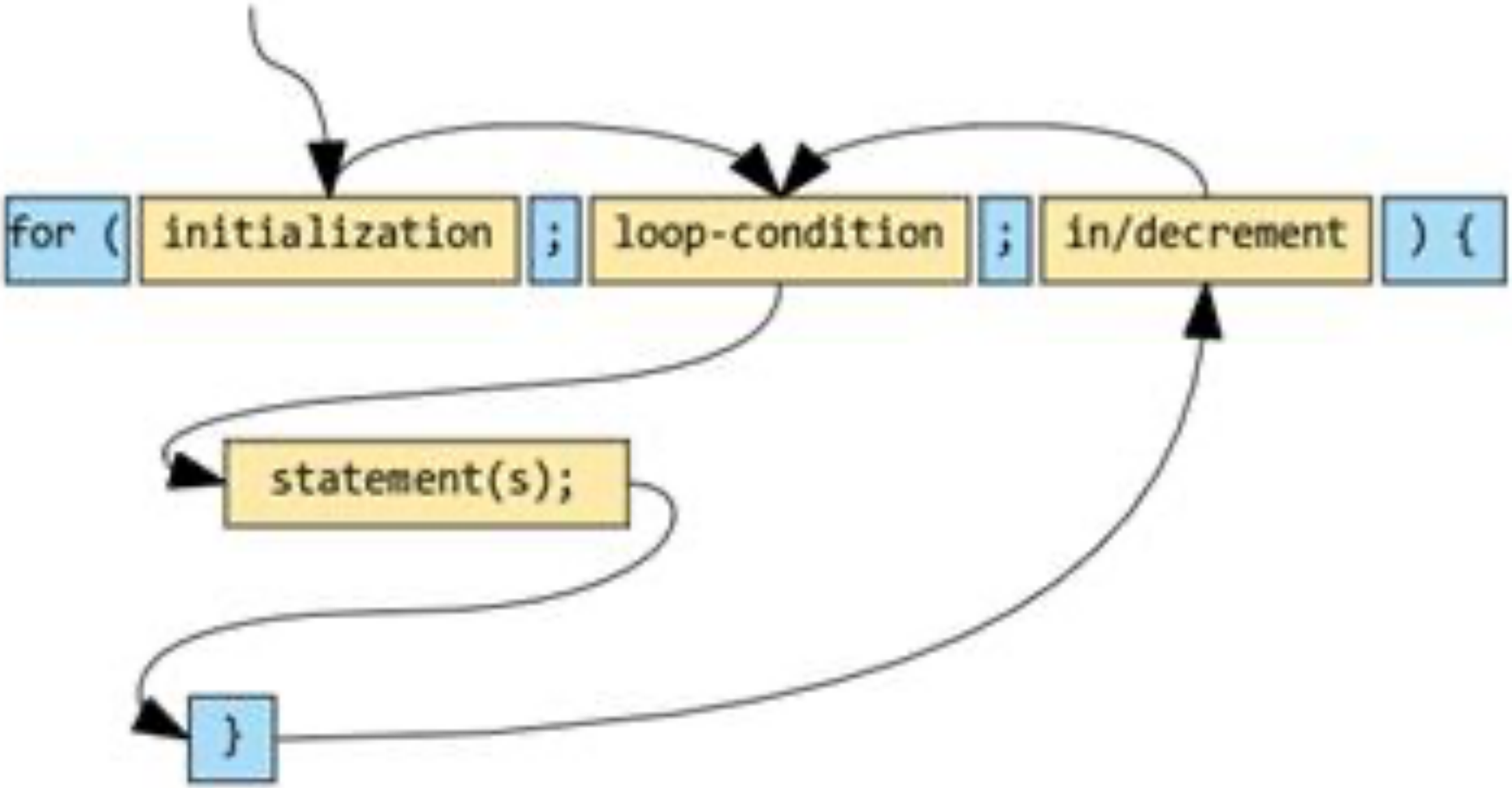
```
// Never-ending loop or infinite loop
while (true) {   // Condition is always true
   // ... Some statements
}
```

While Loop - Controlled

```
int counter = 1;    // Control variable initialized
```

```
// Condition for loop continuation
while (counter <= 10) {
    System.out.println(counter);
    counter++;      // Increment the control variable
}
```

For Loop


for (
initialization
;
loop-condition
;
in/decrement
) {
statement(s);
}

For Loop

```
int counter;
for( counter = 1; counter <= 10; counter++){
   //... Statements
}
```

For Each

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

    int[] arr = { 1, 2, 3, 4, 5 };

    for (int i: arr) {
        System.out.println(i);
    }

}
```

```
1
2
3
4
5
```