TestLeaf

Always Ahead

TestLeaf

Always Ahead

Java - Class

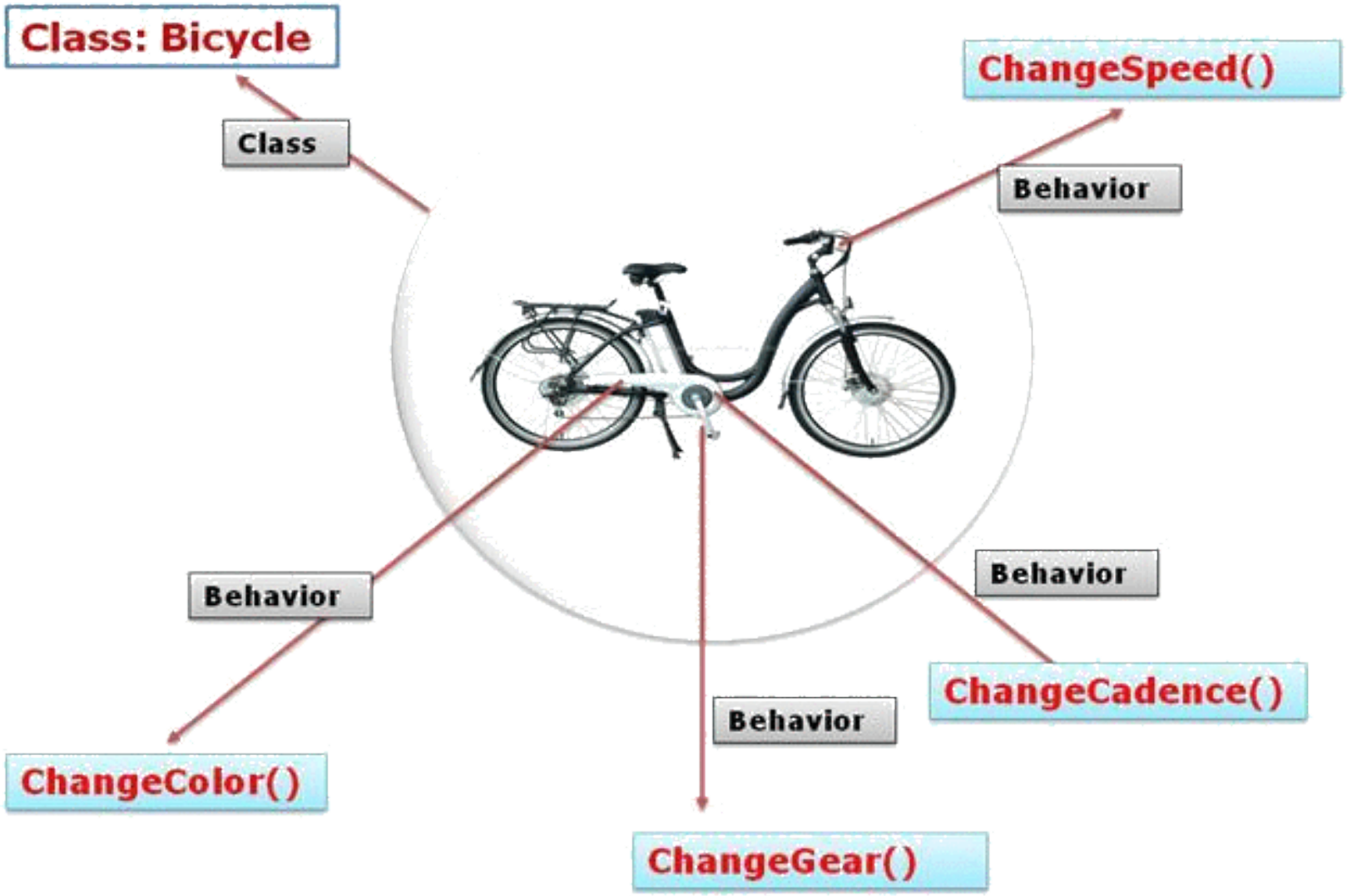
Class

TestLeaf
Always Ahead

Object

Variable

Method



Access Modifier

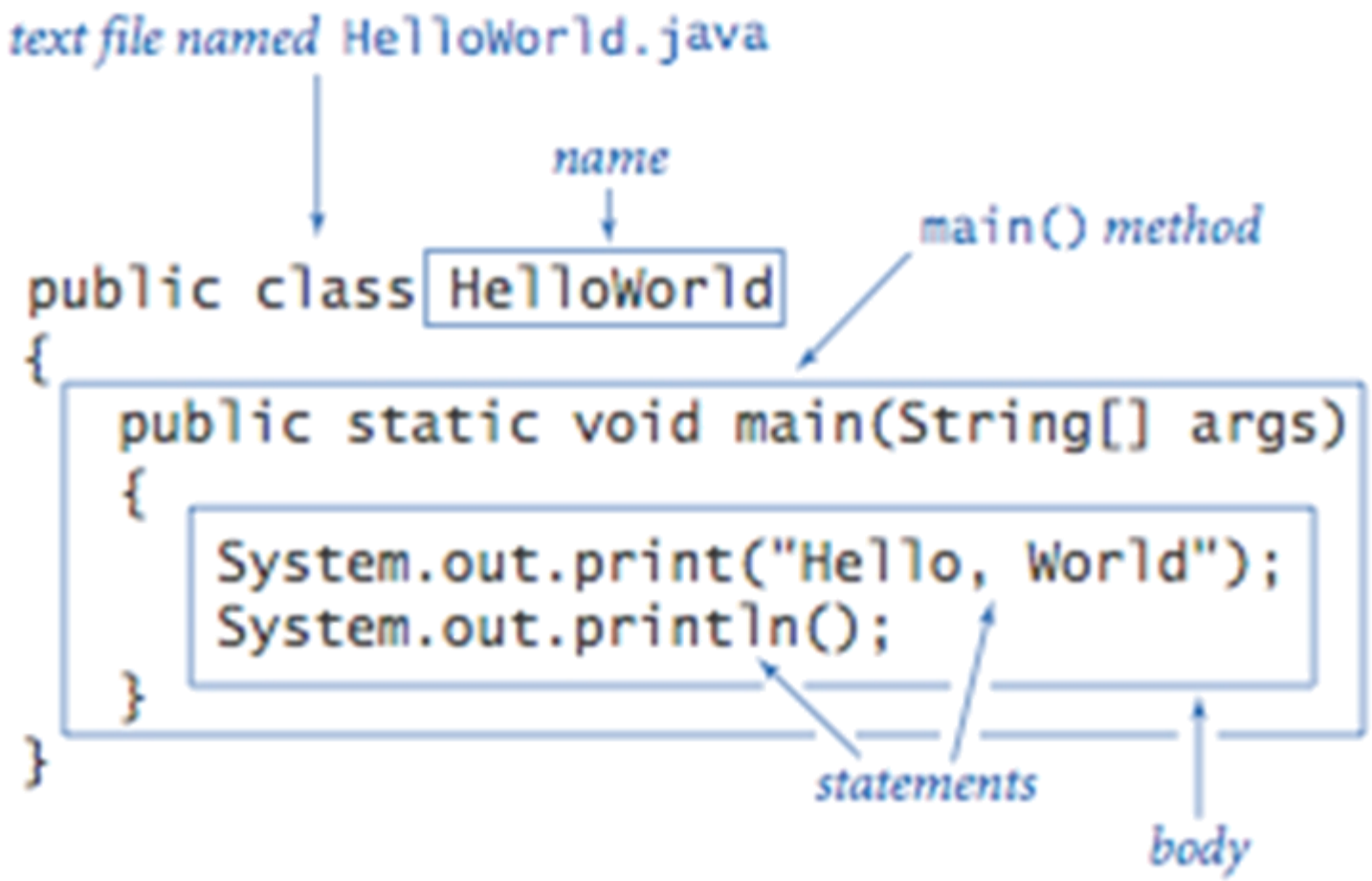
- Class Access Levels – public / no modifier
  - If a class is ‘public’, then it CAN be accessed from ANYWHERE.
  - If a class has ‘no modifier’, then it CAN ONLY be accessed from ‘same package’.

- Member Access Levels – public / protected /no modifier / private
  - public and no modifier – the same way as used in class level.
  - private – members CAN ONLY access.
  - protected – CAN be accessed from ‘same package’ and a subclass existing in any package can access.

Modifier	Same Class	Same Package	Subclass	Other Packages
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Access Levels

Main method

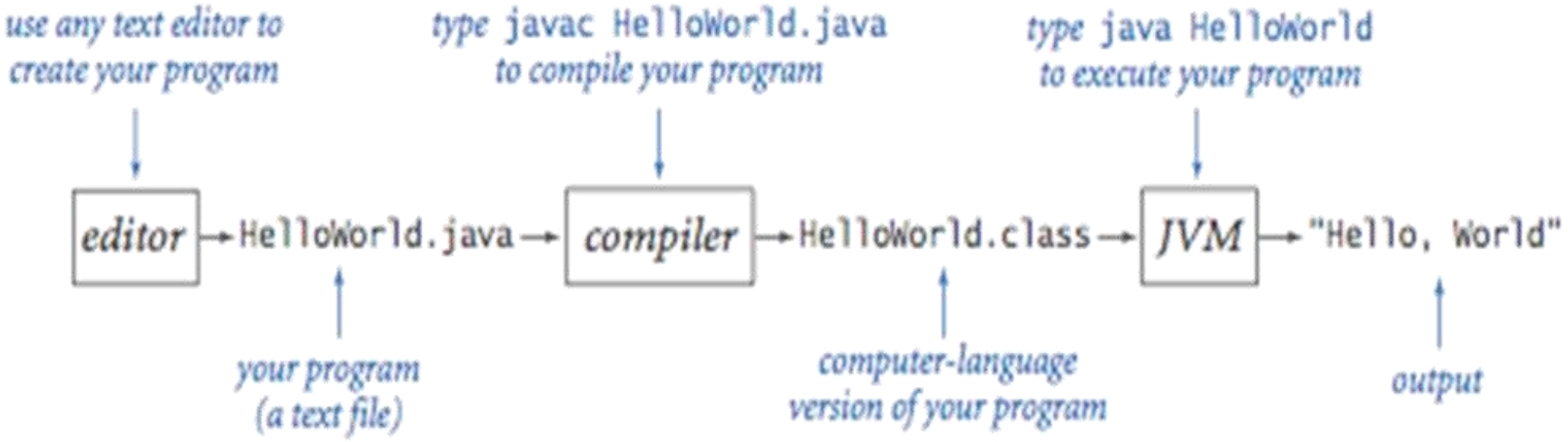


Class Signature

TestLeaf

Always Ahead

Compile & Run

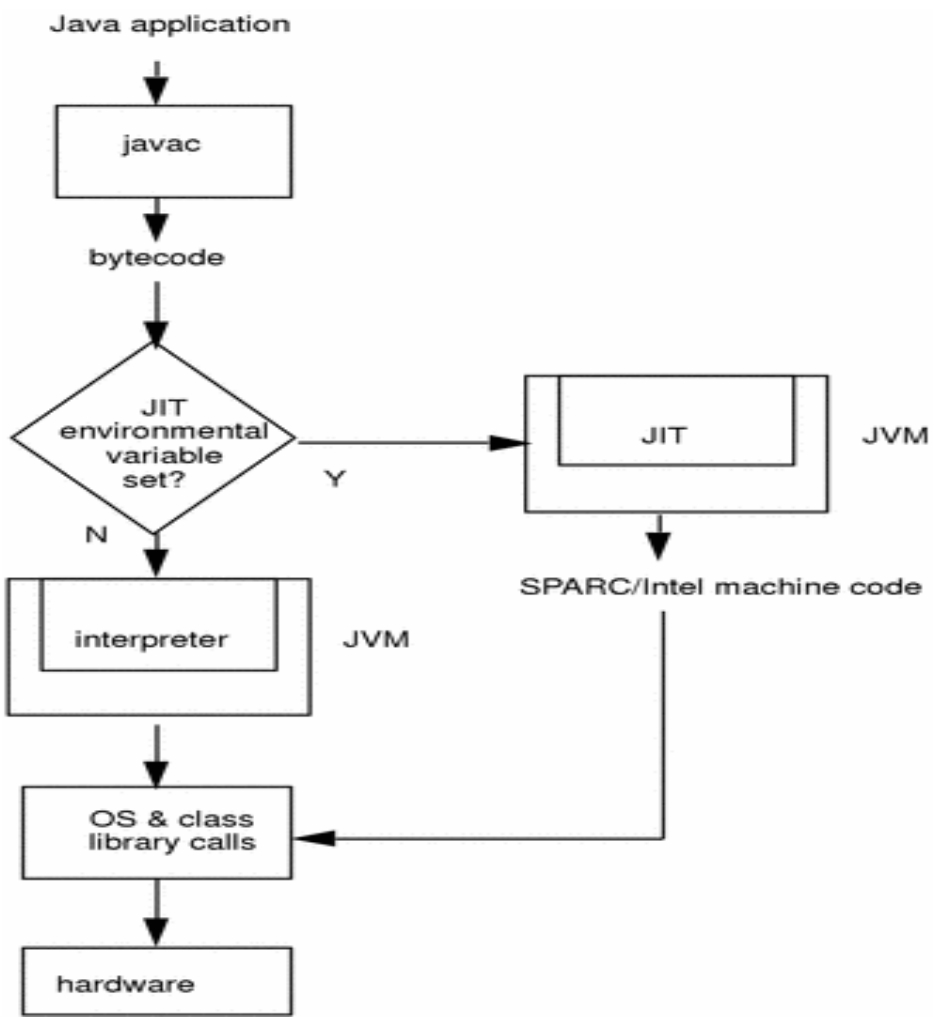


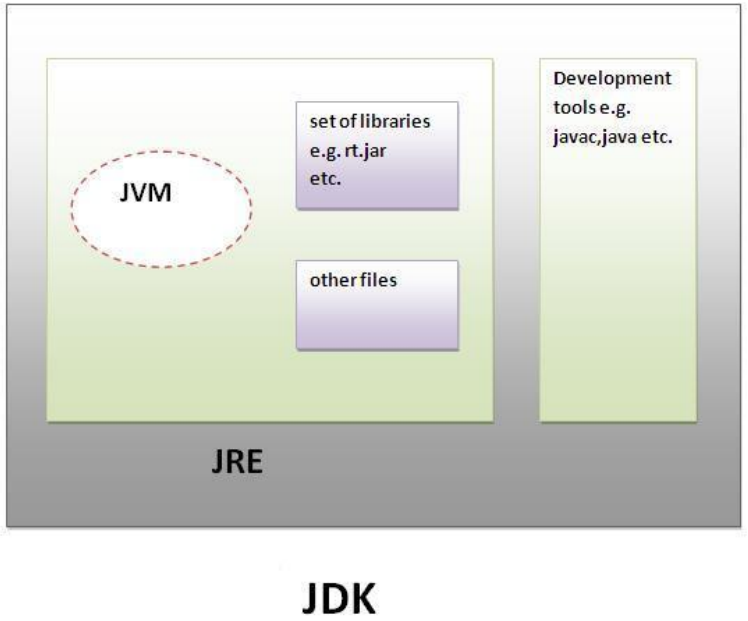
Develop/Edit, Compile, Run

TestLeaf

Always Ahead

JDK vs JRE





Flow of java compilation

http://cs-fundamentals.com/java-programming/difference-between-jdk-jre-jvm-jit.php

TestLeaf

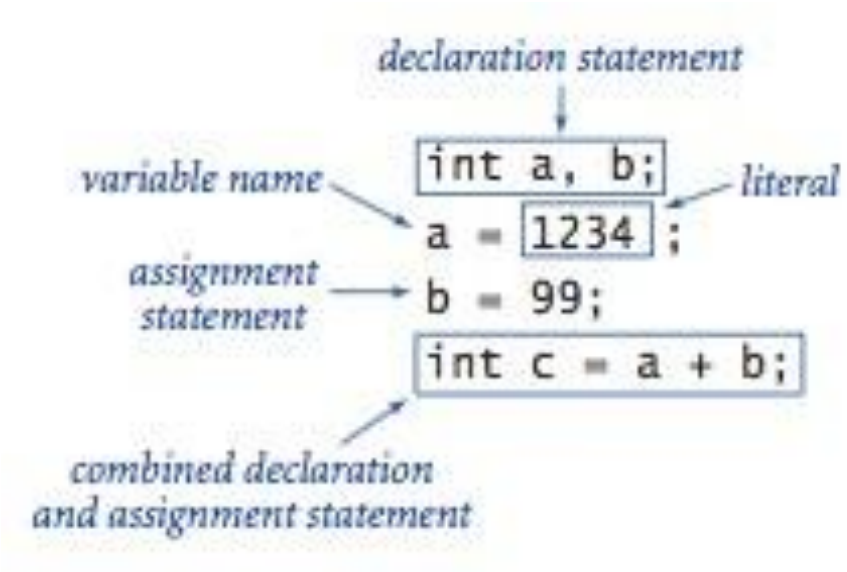
Always Ahead

Data Types

type	*set of values*	*common operators*	*sample literal values*
`int`	integers	`+ - * / %`	`99 -12 2147483647`
`double`	floating-point numbers	`+ - * /`	`3.14 -2.5 6.022e23`
`boolean`	boolean values	`&& || !`	`true false`
`char`	characters		`'A' '1' '%' '\n'`
`String`	sequences of characters	`+`	`"AB" Hello" "2.5"`

Built in Data Types

TestLeaf
Always Ahead



Declaration and assignment statements

values	integers between -2^{31} and $+2^{31}-1$				
typical literals	1234 99 -99 0 1000000				
operations	add	subtract	multiply	divide	remainder
operators	+	-	*	/	%

expression	*value*	*comment*
5 + 3	8	
5 - 3	2	
5 * 3	15	
5 / 3	1	no fractional part
5 % 3	2	remainder
1 / 0		run-time error
3 * 5 - 2	13	* has precedence
3 + 5 / 2	5	/ has precedence
3 - 5 - 2	-4	left associative
(3 - 5) - 2	-4	better style
3 - (5 - 2)	0	unambiguous

Learn - Integers

TestLeaf
Always Ahead

values	real numbers (specified by IEEE 754 standard)			
typical literals	`3.14159`	`6.022e23`	`-3.0` `2.0`	`1.4142135623730951`
operations	add	subtract	multiply	divide
operators	`+`	`-`	`*`	`/`

expression	*value*
`3.141 + .03`	`3.171`
`3.141 - .03`	`3.111`
`6.02e23 / 2.0`	`3.01e23`
`5.0 / 3.0`	`1.6666666666666667`
`10.0 % 3.141`	`0.577`
`1.0 / 0.0`	`Infinity`
`Math.sqrt(2.0)`	`1.4142135623730951`
`Math.sqrt(-1.0)`	`NaN`

Learn – Floating point numbers

TestLeaf
Always Ahead

values	true or false		
literals	`true` `false`		
operations	and	or	not
operators	&&	||	!

a	!a
true	false
false	true

a	b	a && b	a || b
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Learn - Boolean

TestLeaf
Always Ahead

op	meaning	true	false
==	equal	2 == 2	2 == 3
!=	not equal	3 != 2	2 != 2
<	less than	2 < 13	2 < 2
<=	less than or equal	2 <= 2	3 <= 2
>	greater than	13 > 2	2 > 13
>=	greater than or equal	3 >= 2	2 >= 3

Comparison Operators

int	Integer.parseInt(String s)	*convert* s *to an* int *value*
double	Double.parseDouble(String s)	*convert* s *to a* double *value*
long	Long.parseLong(String s)	*convert* s *to a* long *value*

expression	*expression type*	*expression value*
"1234" + 99	String	"123499"
Integer.parseInt("123")	int	123
(int) 2.71828	int	2
Math.round(2.71828)	long	3
(int) Math.round(2.71828)	int	3
(int) Math.round(3.14159)	int	3
11 * 0.3	double	3.3
(int) 11 * 0.3	double	3.3
11 * (int) 0.3	int	0
(int) (11 * 0.3)	int	3

Parsing / Conversion

absolute value	`if (x < 0) x = -x;`

put x and y into sorted order	`if (x > y)`
	`{`
	`  int t = x;`
	`  y = x;`
	`  x = t;`
	`}`

maximum of x and y	`if (x > y) max = x;` `else  max = y;`
error check for division operation	`if (den == 0) System.out.println("Division by zero");` `else  System.out.println("Quotient = " + num/den);`

error check for quadratic formula	`double discriminant = b*b - 4.0*c;`
	`if (discriminant < 0.0)`
	`{`
	`  System.out.println("No real roots");`
	`}`
	`else`
	`{`
	`  System.out.println((-b + Math.sqrt(discriminant))/2.0);`
	`  System.out.println((-b - Math.sqrt(discriminant))/2.0);`
	`}`

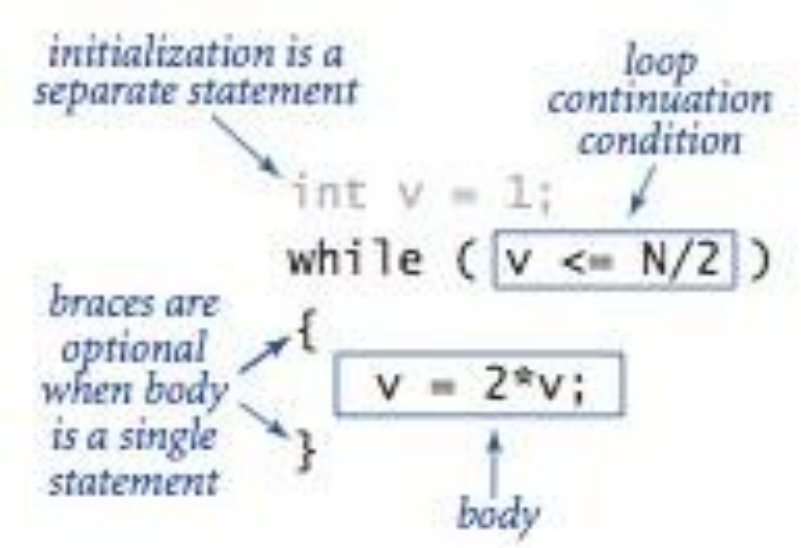
```
if      (income <      0) rate = 0.0;
else if (income <  47450) rate = .22;
else if (income < 114650) rate = .25;
else if (income < 174700) rate = .28;
else if (income < 311950) rate = .33;
else                      rate = .35;
```

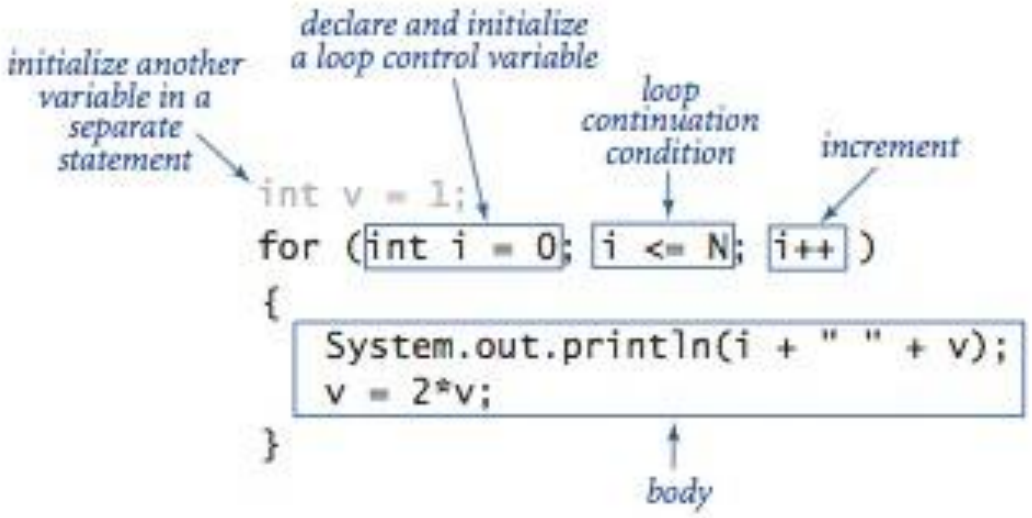
If Else , Nested If

TestLeaf

Always Ahead

Loops





While Vs. For loop

```
int[] numbers = { 10, 20, 30, 40, 50 };
for (int x : numbers) {
        if (x == 30) {
                break;
        }
        System.out.println(x);
}
```

```
int[] numbers = { 10, 20, 30, 40, 50 };
for (int x : numbers) {
        if (x == 30) {
            Continue;
        }
        System.out.println(x);
}
```

Break , Continue statement

```java
switch (day)
{
   case 0: System.out.println("Sun"); break;
   case 1: System.out.println("Mon"); break;
   case 2: System.out.println("Tue"); break;
   case 3: System.out.println("Wed"); break;
   case 4: System.out.println("Thu"); break;
   case 5: System.out.println("Fri"); break;
   case 6: System.out.println("Sat"); break;
}
```

Switch statement