Lecture 1

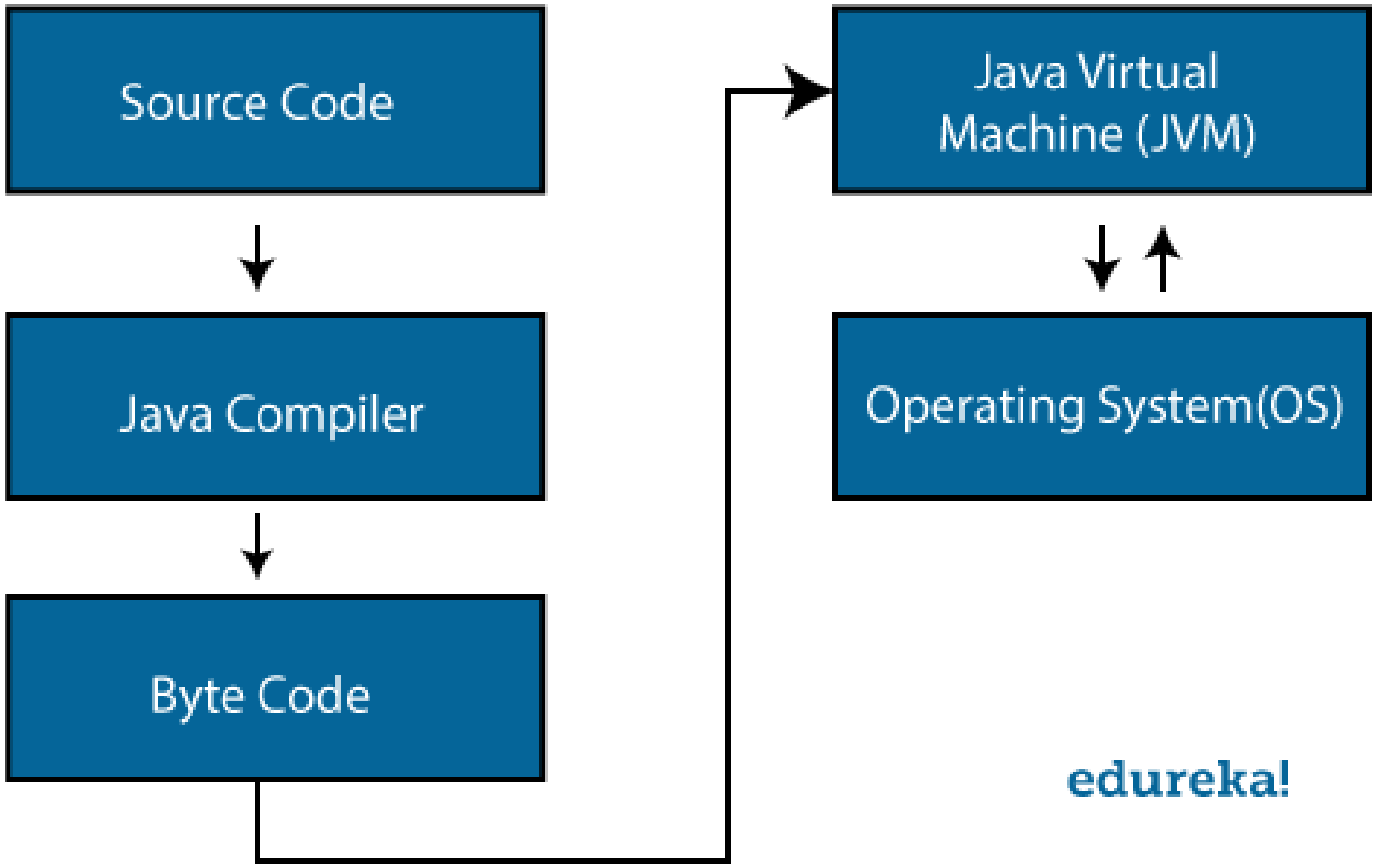
JAVA Review

Teera Siriteerakul

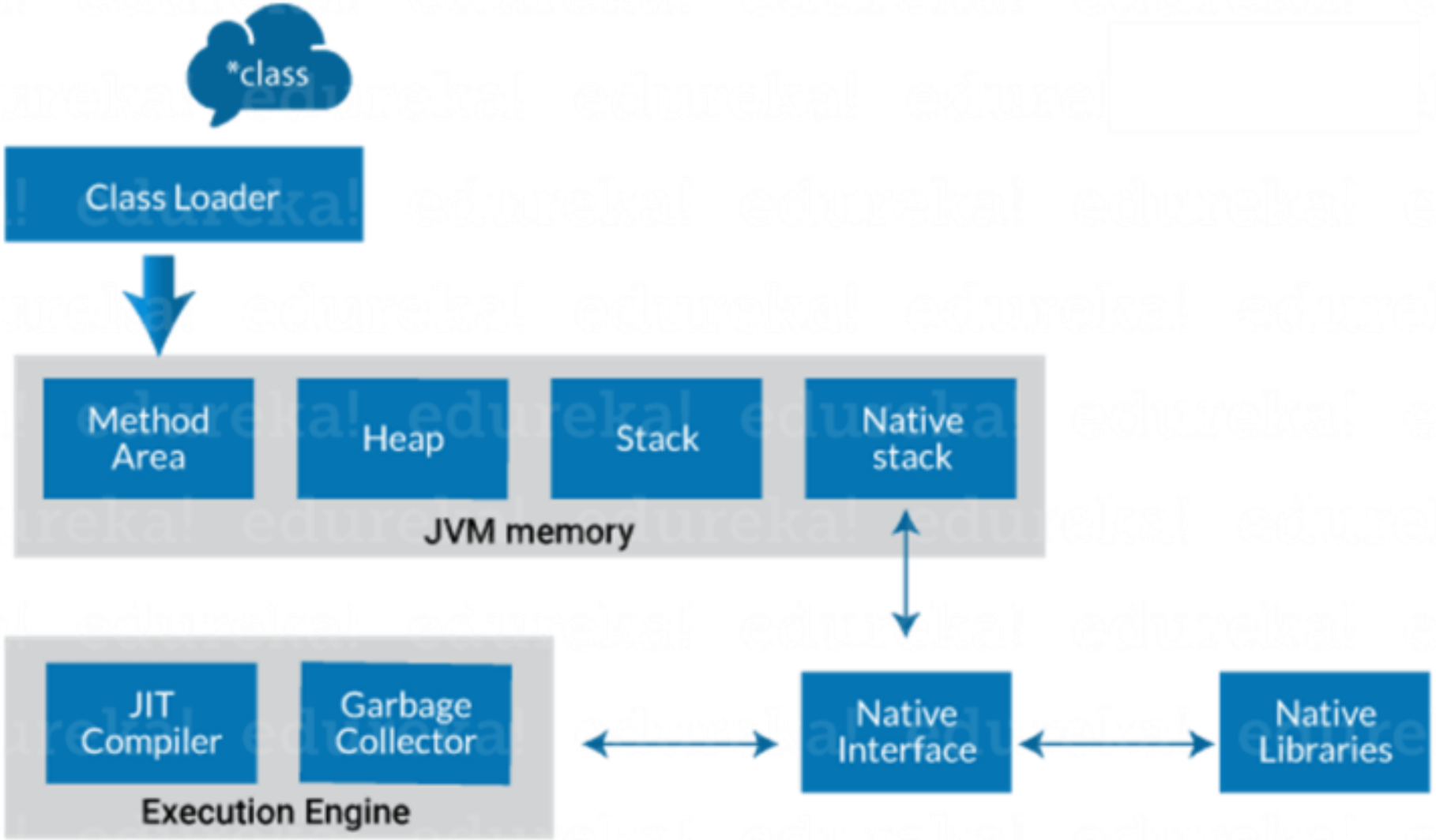
JAVA Application

- Collection of classes
  - One of them have a designated main method.
  - Can be packed into JAVA archive (.jar) and run as an executable (.exe) in JAVA ready machine
- Create by SUN Microsystem
  - Slogan from 1995: Write (and compile) once, run anywhere
  - Bought by Oracle in 2010,
    - Thus, Oracle's implementation (licensed) is the de facto standard.
    - OpenJDK is one of the notable free implementation.

JAVA Programing



JAVA Virtual Machine – JVM



HelloYou.java

```
1  import java.util.Scanner;
2
3  public class HelloYou {
4      public static void main(String args[]) {
5          Scanner in = new Scanner(System.in);
6          System.out.print("Enter your name: ");
7          String yourName = in.nextLine();
8          System.out.println("Hello "+yourName+"!");
9      }
10 }
11
12
```

Some Reserves

boolean	default	for	private	switch
break	double	if	*protected*	this
case	else	import	public	throws
catch	extends	int	return	try
char	final	new	static	void
class	float	*package*	super	while

Built-in 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"`

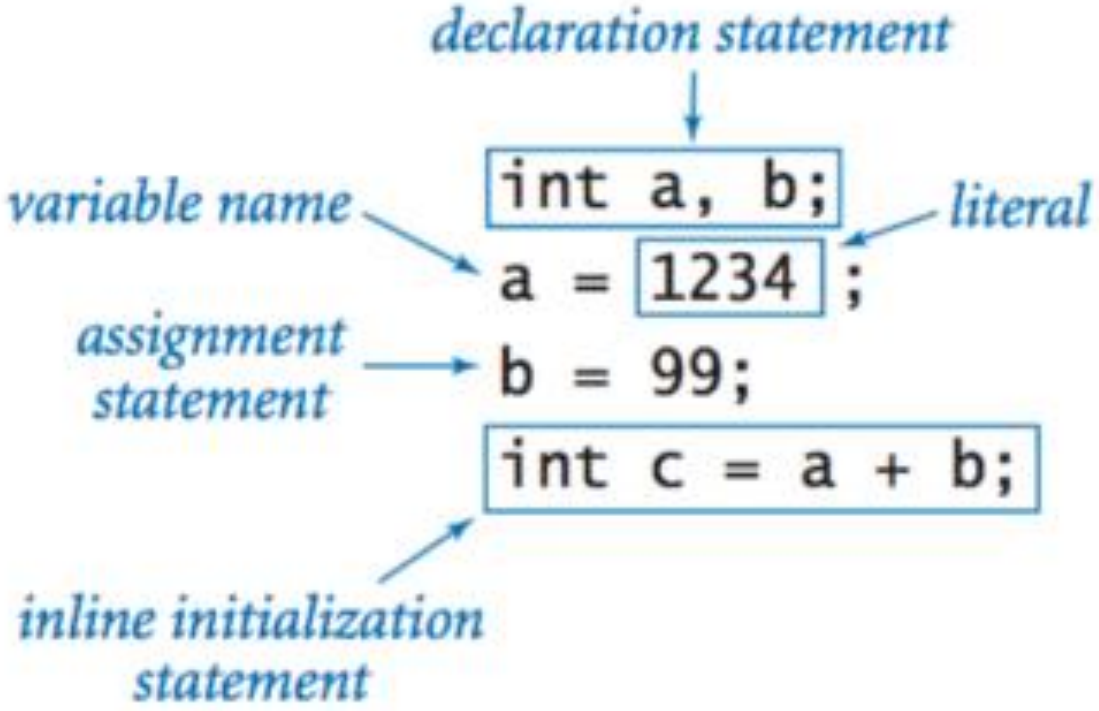
Primitives and Wrapper Classes

- Boolean type
  - boolean → Boolean
- Character type
  - char → Character ***Unicode***
- Integer types
  - byte → Byte
  - short → Short
  - int → Integer
  - long → Long
- Floating point
  - float → Float
  - double → Double

Literals

- boolean
  - {true, false}
- char
  - {'a','b',…}
- int
  - Decimal: 232
  - Octal: 0231 *** leading zero*
  - Hexadecimal: 0X1A *** capital X*
  - Binary: 0b1101
- String
  - "String" *** not primitives*
- Object
  - null

Declaration and assignment



Operators

Integers.

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

Floating-point numbers.

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

Booleans.

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

Comparison Operators

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`

String

public class String

	String(String s)	*create a string with the same value as* `s`
	String(char[] a)	*create a string that represents the same sequence of characters as in* `a[]`
int	length()	*number of characters*
char	charAt(int i)	*the character at index* `i`
String	substring(int i, int j)	*characters at indices* `i` *through* `(j-1)`
boolean	contains(String substring)	*does this string contain* `substring`*?*
boolean	startsWith(String prefix)	*does this string start with* `prefix`*?*
boolean	endsWith(String postfix)	*does this string end with* `postfix`*?*
int	indexOf(String pattern)	*index of first occurrence of* `pattern`
int	indexOf(String pattern, int i)	*index of first occurrence of* `pattern` *after* `i`
String	concat(String t)	*this string, with* `t` *appended*
int	compareTo(String t)	*string comparison*
String	toLowerCase()	*this string, with lowercase letters*
String	toUpperCase()	*this string, with uppercase letters*
String	replace(String a, String b)	*this string, with* `a`*s replaced by* `b`*s*
String	trim()	*this string, with leading and trailing whitespace removed*
boolean	matches(String regexp)	*is this string matched by the regular expression?*
String[]	split(String delimiter)	*strings between occurrences of* `delimiter`
boolean	equals(Object t)	*is this string's value the same as* `t`*'s?*
int	hashCode()	*an integer hash code*

Special Characters

Special characters	*Display*
\'	Single quotation mark
\"	Double quotation mark
\\	Backslash
\t	Tab
\b	Backspace
\r	Carriage return
\f	Formfeed
\n	Newline

Output

Printing.

`void System.out.print(String s)`	*print s*
`void System.out.println(String s)`	*print s, followed by a newline*
`void System.out.println()`	*print a newline*

Parsing from String to number

Parsing command-line arguments.

`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*

Math Library

`public class Math`

`double abs(double a)`	*absolute value of a*
`double max(double a, double b)`	*maximum of a and b*
`double min(double a, double b)`	*minimum of a and b*
`double sin(double theta)`	*sine of theta*
`double cos(double theta)`	*cosine of theta*
`double tan(double theta)`	*tangent of theta*
`double toRadians(double degrees)`	*convert angle from degrees to radians*
`double toDegrees(double radians)`	*convert angle from radians to degrees*
`double exp(double a)`	*exponential* (e^a)
`double log(double a)`	*natural log* ($\log_e a$, *or* $\ln a$)
`double pow(double a, double b)`	*raise a to the bth power* (a^b)
`long round(double a)`	*round a to the nearest integer*
`double random()`	*random number in* [0, 1)
`double sqrt(double a)`	*square root of a*
`double E`	*value of e (constant)*
`double PI`	*value of π (constant)*

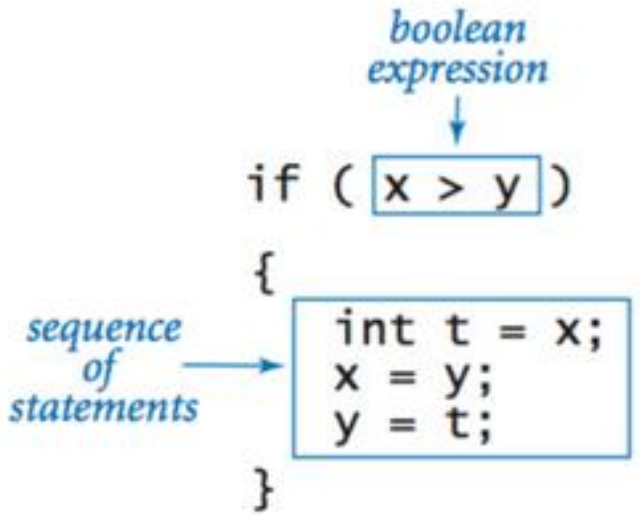
Type Conversion

expression	*expression type*	*expression value*
(1 + 2 + 3 + 4) / 4.0	double	2.5
Math.sqrt(4)	double	2.0
"1234" + 99	String	"123499"
11 * 0.25	double	2.75
(int) 11 * 0.25	double	2.75
11 * (int) 0.25	int	0
(int) (11 * 0.25)	int	2
(int) 2.71828	int	2
Math.round(2.71828)	long	3
(int) Math.round(2.71828)	int	3
Integer.parseInt("1234")	int	1234

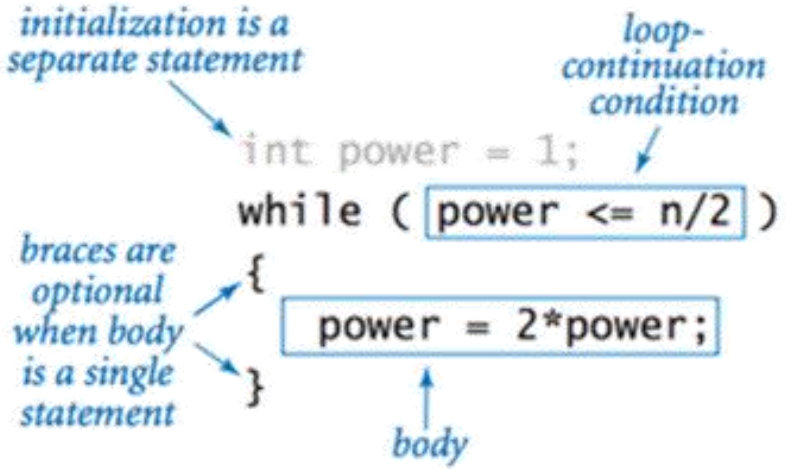
Implicit type casting

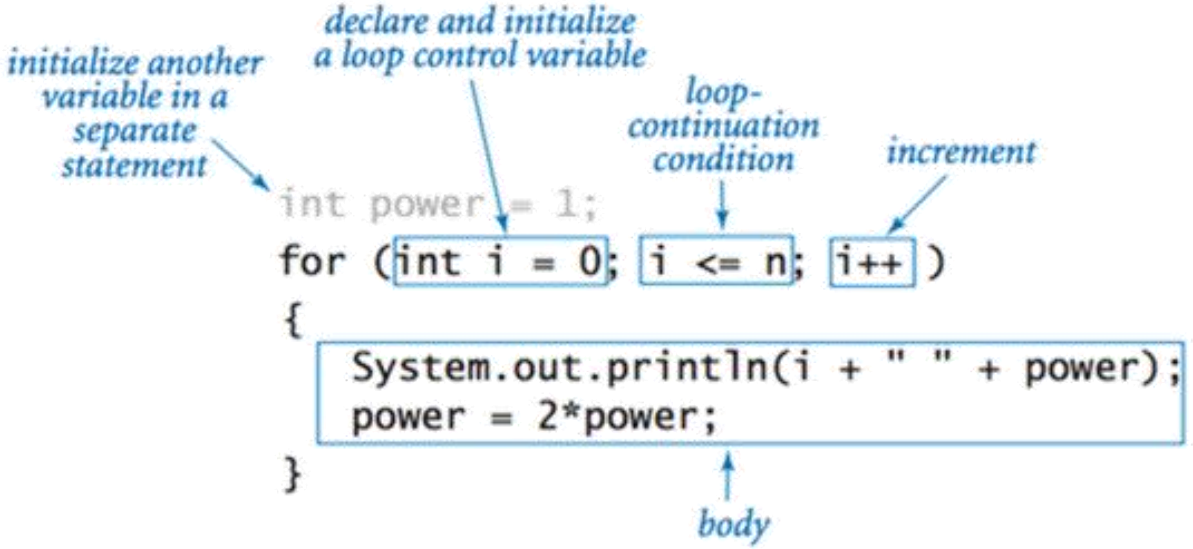
- `int` → `long` → `float` → `double`
- `Ex: double d = 10; // cast int to double`

Condition


boolean expression
if (x > y)
{
int t = x;
x = y;
y = t;
}
sequence of statements

Loops


initialization is a separate statement
loop-continuation condition
int power = 1;
while (power <= n/2)
{
power = 2*power;
}
braces are optional when body is a single statement
body


initialize another variable in a separate statement
declare and initialize a loop control variable
loop-continuation condition
increment
int power = 1;
for (int i = 0; i <= n; i++)
{
System.out.println(i + " " + power);
power = 2*power;
}
body

Do-while loop and break

```
do
{  // Scale x and y to be random in (-1, 1).
   x = 2.0*Math.random() - 1.0;
   y = 2.0*Math.random() - 1.0;
} while (Math.sqrt(x*x + y*y) > 1.0);
```

```
int factor;
for (factor = 2; factor <= n/factor; factor++)
   if (n % factor == 0) break;

if (factor > n/factor)
   System.out.println(n + " is prime");
```

Switch Statement

```
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;
}
```

Array

Inline array initialization.

```
String[] SUITS = { "Clubs", "Diamonds", "Hearts", "Spades" };

String[] RANKS = {
   "2", "3", "4", "5", "6", "7", "8", "9", "10",
   "Jack", "Queen", "King", "Ace"
};
```

Typical array-processing code

create an array with random values	`double[] a = new double[n];`
	`for (int i = 0; i < n; i++)`
	`  a[i] = Math.random();`

print the array values, one per line	`for (int i = 0; i < n; i++)` `  System.out.println(a[i]);`

find the maximum of the array values	`double max = Double.NEGATIVE_INFINITY;`
	`for (int i = 0; i < n; i++)`
	`  if (a[i] > max) max = a[i];`

compute the average of the array values	`double sum = 0.0;`
	`for (int i = 0; i < n; i++)`
	`  sum += a[i];`
	`double average = sum / n;`

reverse the values within an array	`for (int i = 0; i < n/2; i++)`
	`{`
	`  double temp = a[i];`
	`  a[i] = a[n-1-i];`
	`  a[n-i-1] = temp;`
	`}`

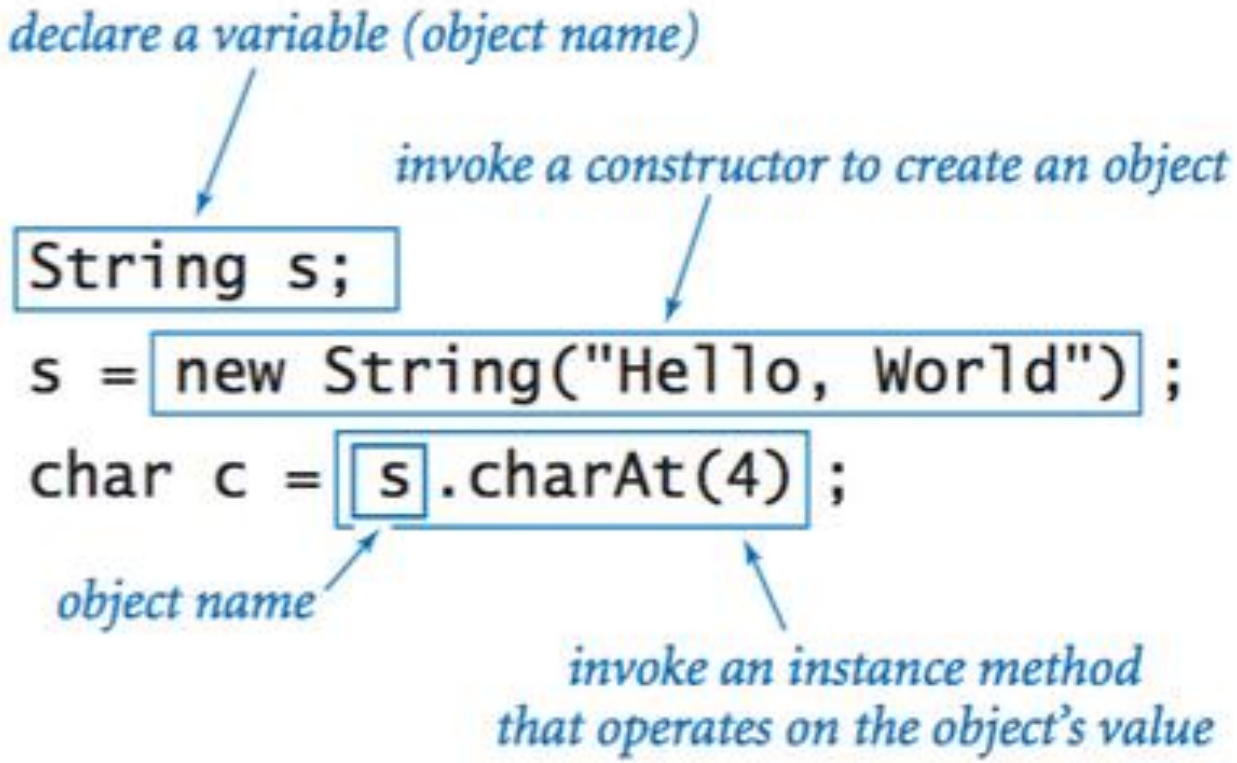
copy sequence of values to another array	`double[] b = new double[n];`
	`for (int i = 0; i < n; i++)`
	`  b[i] = a[i];`

2D Array

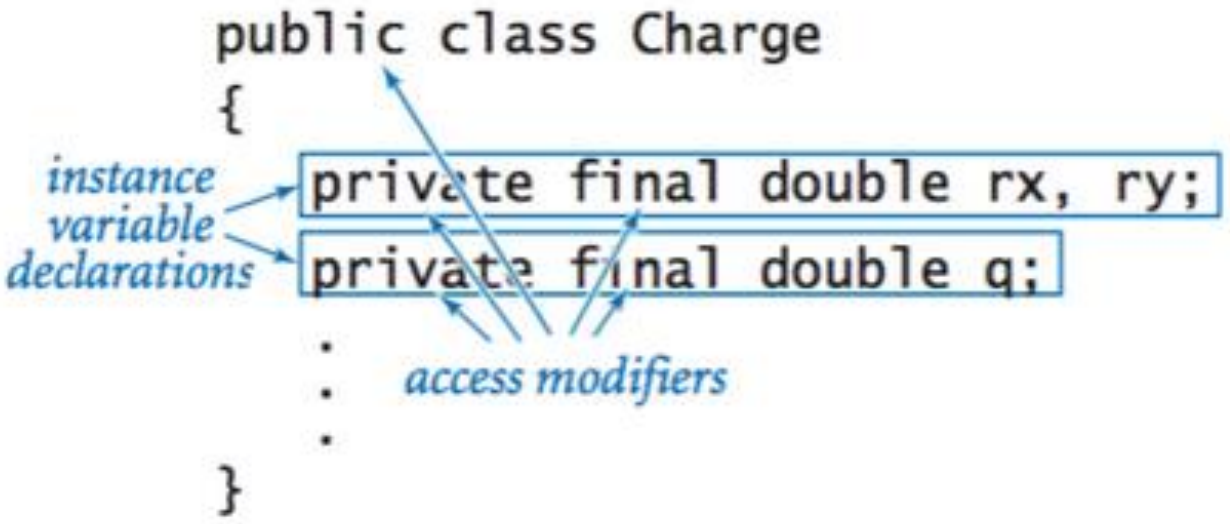
Inline initialization.

```
double [][] a =
{
   { 99.0, 85.0, 98.0,  0.0 },
   { 98.0, 57.0, 79.0,  0.0 },
   { 92.0, 77.0, 74.0,  0.0 },
   { 94.0, 62.0, 81.0,  0.0 },
   { 99.0, 94.0, 92.0,  0.0 },
   { 80.0, 76.5, 67.0,  0.0 },
   { 76.0, 58.5, 90.5,  0.0 },
   { 92.0, 66.0, 91.0,  0.0 },
   { 97.0, 70.5, 66.5,  0.0 },
   { 89.0, 89.5, 81.0,  0.0 },
   {  0.0,  0.0,  0.0,  0.0 }
};
```

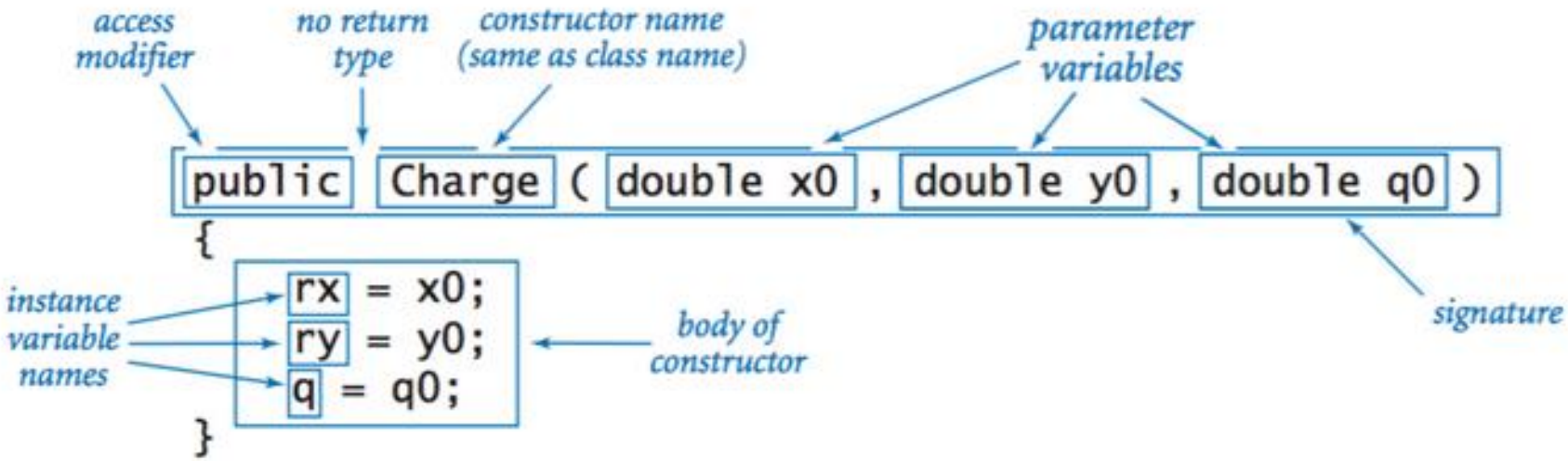
Using an Object



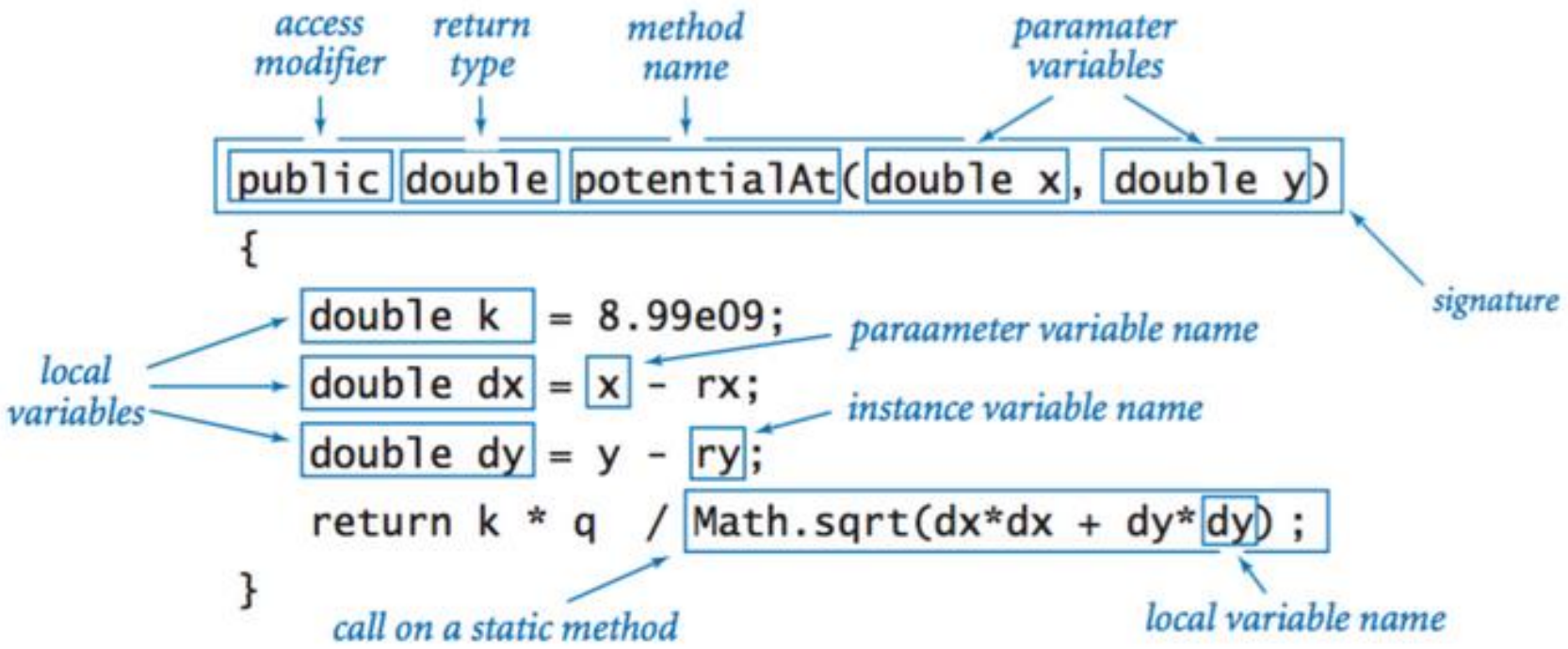
Instance Variable



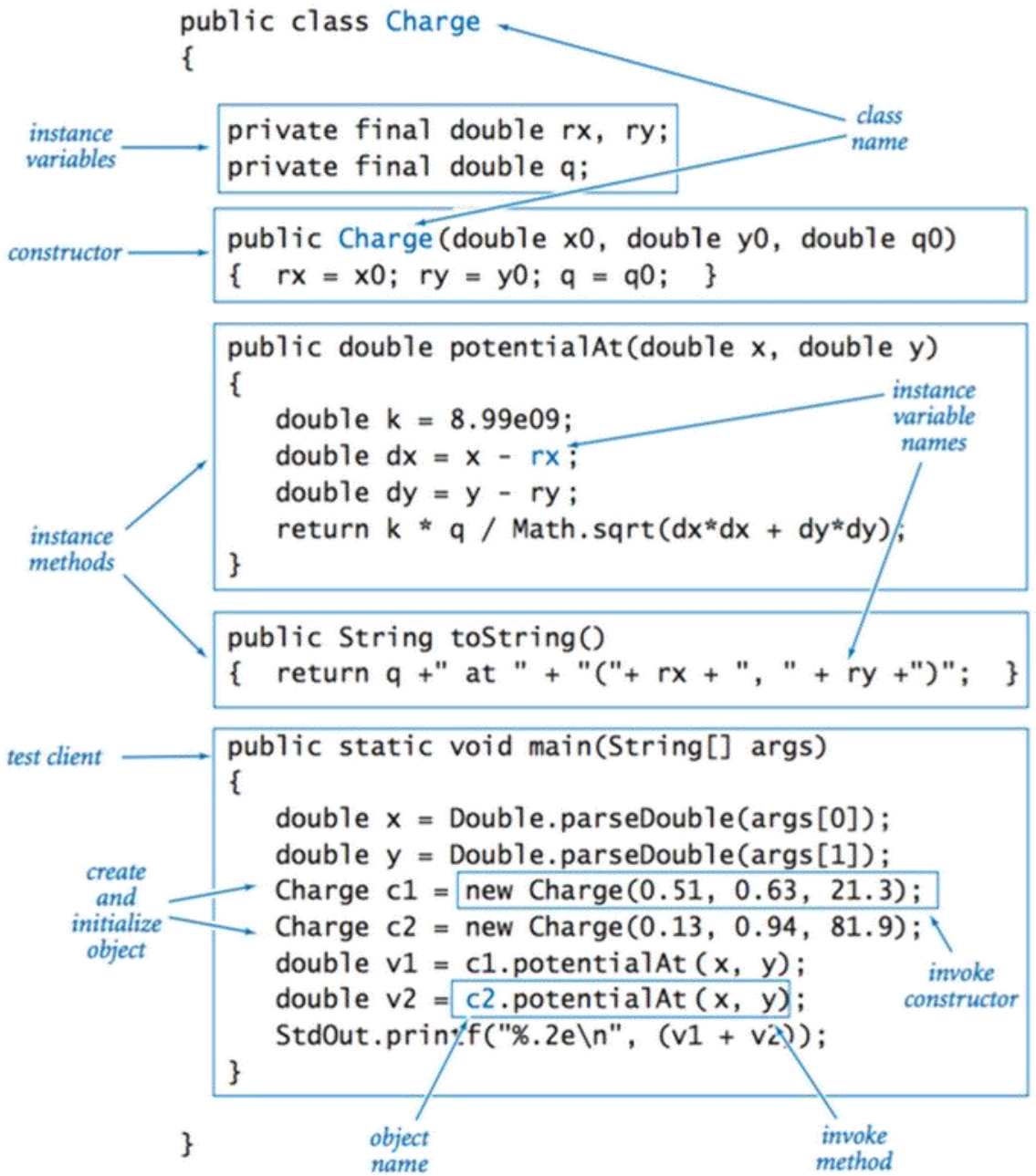
Object Constructors



Instance Methods



Classes


public class Charge
{
private final double rx, ry;
private final double q;
public Charge(double x0, double y0, double q0)
{ rx = x0; ry = y0; q = q0; }
public double potentialAt(double x, double y)
{
double k = 8.99e09;
double dx = x - rx;
double dy = y - ry;
return k * q / Math.sqrt(dx*dx + dy*dy);
}
public String toString()
{ return q +" at " + "("+ rx + ", " + ry +")"; }
public static void main(String[] args)
{
double x = Double.parseDouble(args[0]);
double y = Double.parseDouble(args[1]);
Charge c1 = new Charge(0.51, 0.63, 21.3);
Charge c2 = new Charge(0.13, 0.94, 81.9);
double v1 = c1.potentialAt(x, y);
double v2 = c2.potentialAt(x, y);
StdOut.printf("%.2e\n", (v1 + v2));
}
}
instance variables
class name
constructor
instance variable names
instance methods
test client
create and initialize object
invoke constructor
object name
invoke method

Access Modifier

- private
- protected
- public
- Used for classes, variables, and methods

Ternary Operator (Conditional Operator)

Setting max

- Condition statement

```
if (a>b) {
    max = a;
} else {
    max = b;
}
```

- Ternary Operator

```
max = (a>b)?a:b;
```

References

- https://introcs.cs.princeton.edu/java/11cheatsheet/
- https://www.upgrad.com/blog/types-of-literals-in-java/
- http://www2.hawaii.edu/~tp_200/lectureNotes/review_of_some_java_basics.htm
- http://comet.lehman.cuny.edu/sfakhouri/teaching/cmp/cmp338/lecturenotes-3rdEdition/Chapter-01.pdf