Programming Languages:

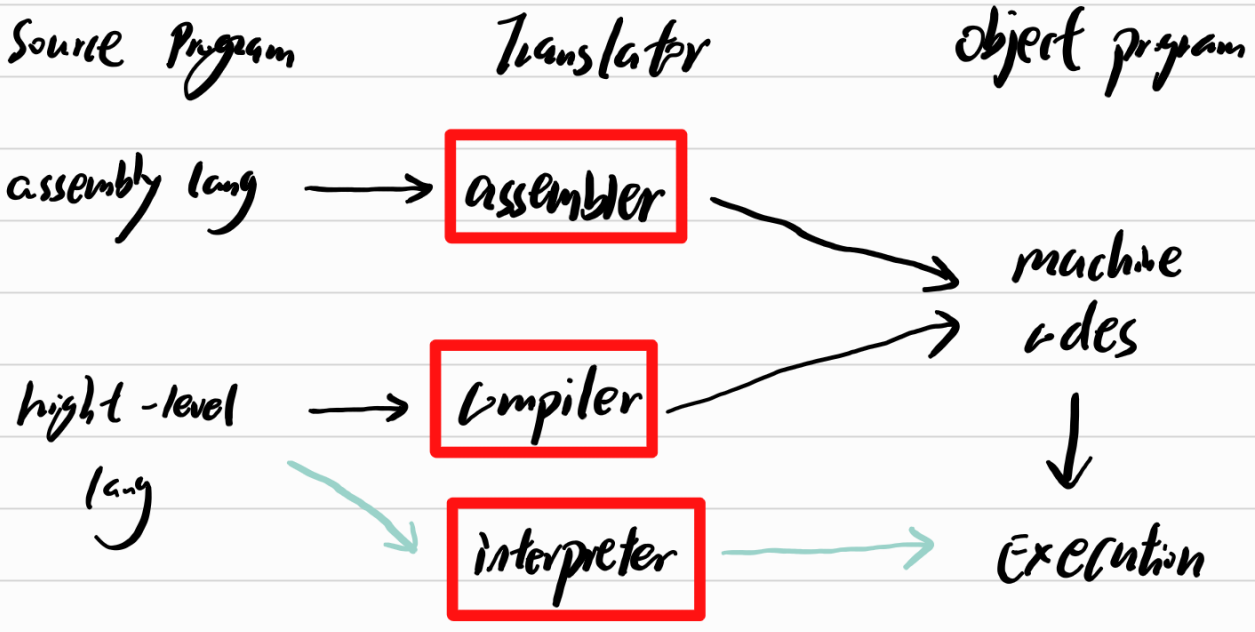
problem → Program (input, Alg, output) → solution

- Machine Language (Native)
- Assembly Language
- High-level Language

```
public   abstract   class   GrossPay { }
```

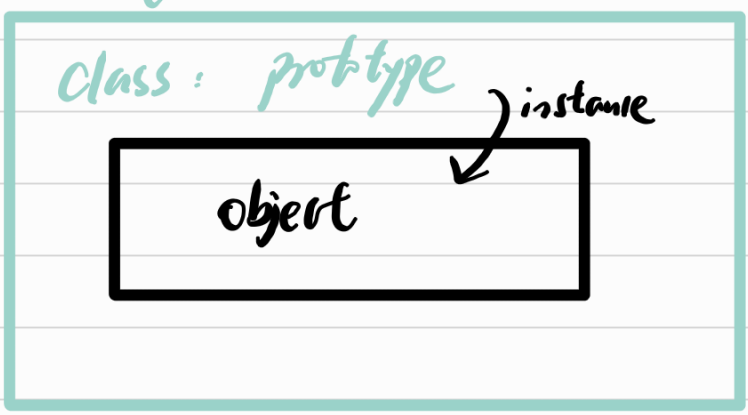
- public: access modifier
- abstract: make it abstract
- class: keyword "class"
- GrossPay: Name

→ Translators:



H-L langs { Procedural langs

obj-oriented langs



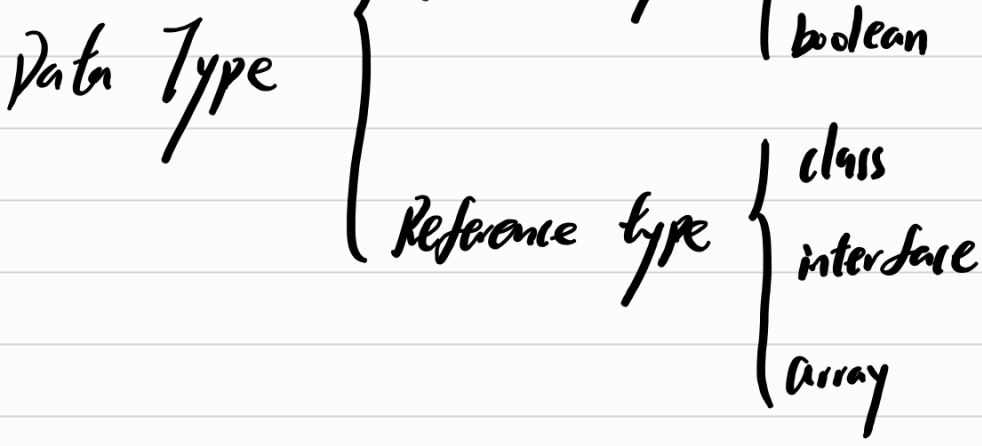
- An object is an instance of a class
- A class is a blueprint from which objects are made.

object-oriented
- ① Do not Repeat Yourself
- ② Should not reveal the inf about itself unless necessary
- ③ Send a message rather than doing a function call.

Benefits
- modularity
- information-hiding
- code re-use

Basic types in Java are called primitive types

primitive type
- Numerical
  - integer: byte, short, int, long
  - float: float, double
- char



Data Type	Default Value (for fields)	Size (in bits)	Minimum Range	Maximum Range
byte	0	Occupy 8 bits in memory	-128	+127
short	0	Occupy 16 bits in memory	-32768	+32767
int	0	Occupy 32 bits in memory	-2147483648	+2147483647
long	0L	Occupy 64 bits in memory	-9223372036854775808	+9223372036854775807
float	0.0f	Occupy 32-bit IEEE 754 floating point	1.40129846432481707e-45	3.40282346638528860e+38
double	0.0d	Occupy 64-bit IEEE 754 floating point	4.94065645841246544e-324d	1.79769313486231570e+308d
char	'\u0000'	Occupy 16-bit, unsigned Unicode character		0 to 65,535
boolean	false	Occupy 1- bit in memory	NA	NA

floating-point number literals are considered to be double.
numbers w/ decimal point
numbers in exponential notation

Octal number begin with "0"
Hexadecimal begin with "0X" or "0x"

```
Person person1 = new Person();
```

[<modifier>] type <fields_name> [= default Value]

Java does nt have global variable

method overload : in same class name

can have dif parameter list
return type

Modifier	same class	same package	subclass	Other place
`private`	Yes	No	No	No
no modifier	Yes	Yes	No	No
`protected`	Yes	Yes	Yes	No
`public`	Yes	Yes	Yes	Yes

Only public and no modifier can be used in class declaration

Inheritance
- method override
- super
- this
- final
- abstract
- static

tString() equals(object O)

Dynamic Binding

- Override : in subclass
  same name & signature
- Overload : in same class
  same name
  dif signature

Subtype Polymorphism: an obj from a subclass can be used as if it were an obj. from a superclass.
(freely upcast)

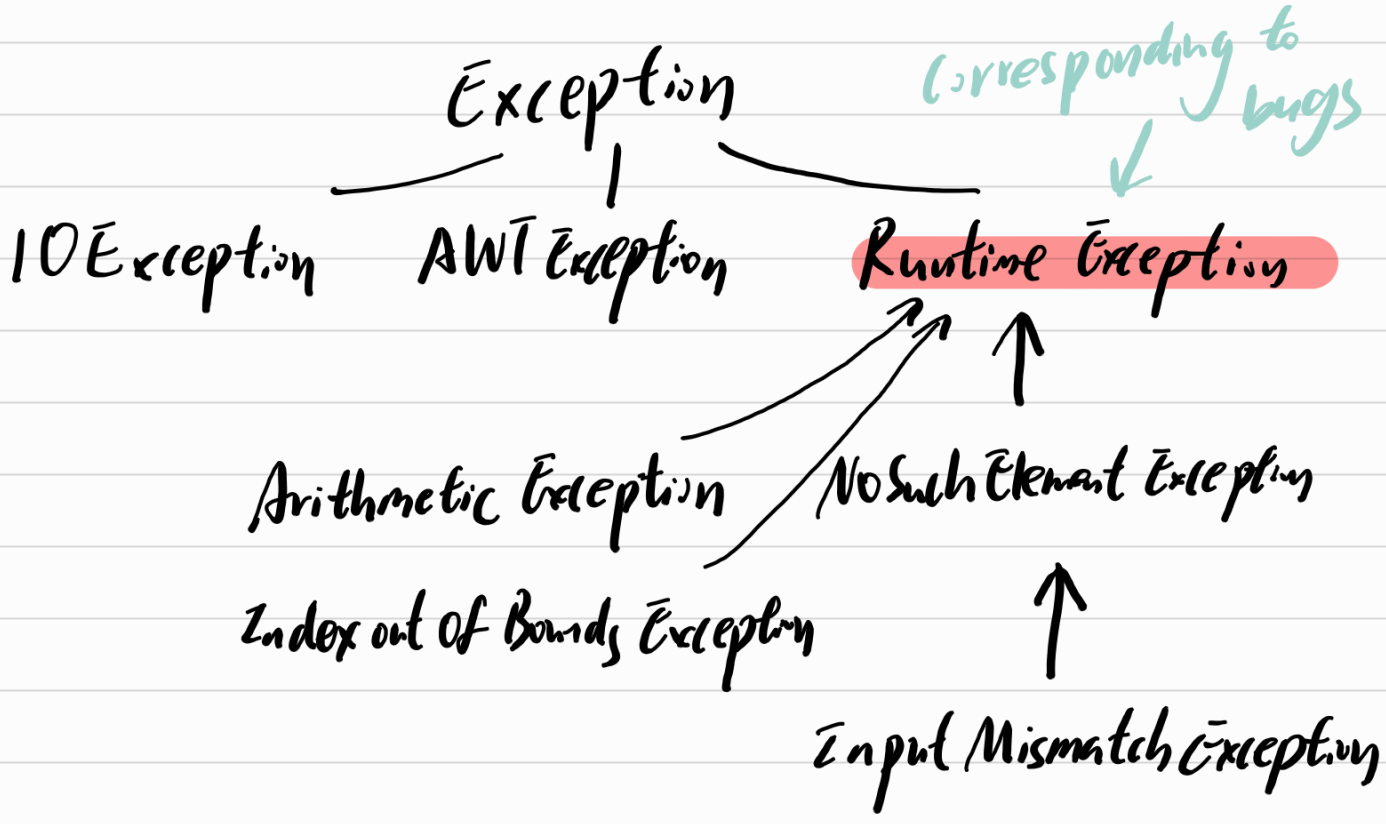
downcast: only works if an object satisfied
i.e.
- all prerequisite the destincted class.
- (only works if an obj. is downcasted back to its original type after upcast)

object instanceof class

Exception Handling :

- Compile-time errors: lexical / syntactic / semantic
- Runtime errors

Java is safe:
① automatically detect that sth. wrong
② immediately stop the program
③ Show you an exception about the prob
④ tell you what the exception is about
⑤ tell you exactly where the exception happened in your code.



- java.lang
- java.awt
- java.swing
- java.net

java.io

java.util

```
try {

} + at least 1 catch() or finally()

catch (<exception type>) {

}

catch (<exception type>) {

}

finally {

}
```

Integer.parseInt()

getMessage()

↳return the detailed error msg string for this exception object.

printStackTrace()

↳ prints this exception and its backtrace (the list of method calls starting from main up to the method that threw the exception object.)

```
public static void readPosInt() throws Exception{
        Scanner scanner = new Scanner(System.in);
        System.out.print("input a positive int: ");
        String s = scanner.nextLine();
        int i = Integer.parseInt(s);

        if(i <= 0){
            throw new Exception("i <= 0 !");
        }
        System.out.println("i is: " + i);
}
```

What if some exception never caught:

① in non-GUI program: the exception will kill everything and show the caught exception on screen

② in GUI proj: only kill the thread with exception and throw a exception

throws @Override: its override methods can only be the same or child exceptions

Array: int[] a = new int[n];

length: a.length.

⇒ enhanced for l.p:

(primitive/class) element type | VarName | arrayName

```
for (Student s: a) {
}
```

object ← array

readonly for iteration

ArrayList can be dynamically adding or removing

→ the initial size of an arraylist is 0.

→ by default, the type of elements

of an ArrayList is Object.

(≈ required downcast).

⇒ add(), get(i), set(index, val)

⇒ int → Integer

(Integer → int: .intValue())

double → Double

boolean → Boolean

- boxing: automatic conversion
- when take out the obj from ArrayList, Java can automatically unbox

Generics : a way to parameterize a class over a type

- parametric polymorphism
- ad-hoc polymorphism (overloading)
- subtype polymorphism

```
public class Box <T> {

    private T data;

    public Box (T data){
        this.data = data;
```

```
}
public T getData() {
    return this.data;
}
public void setData(T data) {
    this.data = data;
} ...
```