Java Review

Sophia Tao, Jan 21, 2020

Disclaimer

- This slideset does not include all the content in the quiz
- To study, go through all the labs
- Not just the code, but the explanations too

Quiz is worth **6%**

Java is...

- Object-oriented
- Statically-typed
- Still the most used language as of 2019 (https://www.tiobe.com/tiobe-index/)

Everything is a class in Java!

Control flows in Java

if statements

while statements

for statements

```
if (a < b) {
    a -= 1;
}

while (a < b) {
    a -= 1;
}

for (int i = 0; i < 10; i++) {
    a -= 1;
}
```

Types in Java

Primitive:

```
int a = 1; // integer
boolean b = true; // true or false
double c = 1.8; // decimals
char d = '&'; // characters
```

Non-primitive:

```
String e = "String Example";
YourClassHere f = new YourClassHere();
```

Java can cast one type to another

Explicit casting

```
int integer = (int) 1.0;
```

Implicit casting

```
int integer = 1;
String string = "Hello" + integer; // Hello1
```

Arrays in Java

```
int size = 5;
int[] a = new int[size];
String[] e = new String[4];
```

Keywords for variables or functions

ACCESS: public / private / package / _

STATICNESS: static / _

FINALNESS: final / _

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

For functions only:

RETURN TYPE: void / int / double / String / YourClassHere

Static

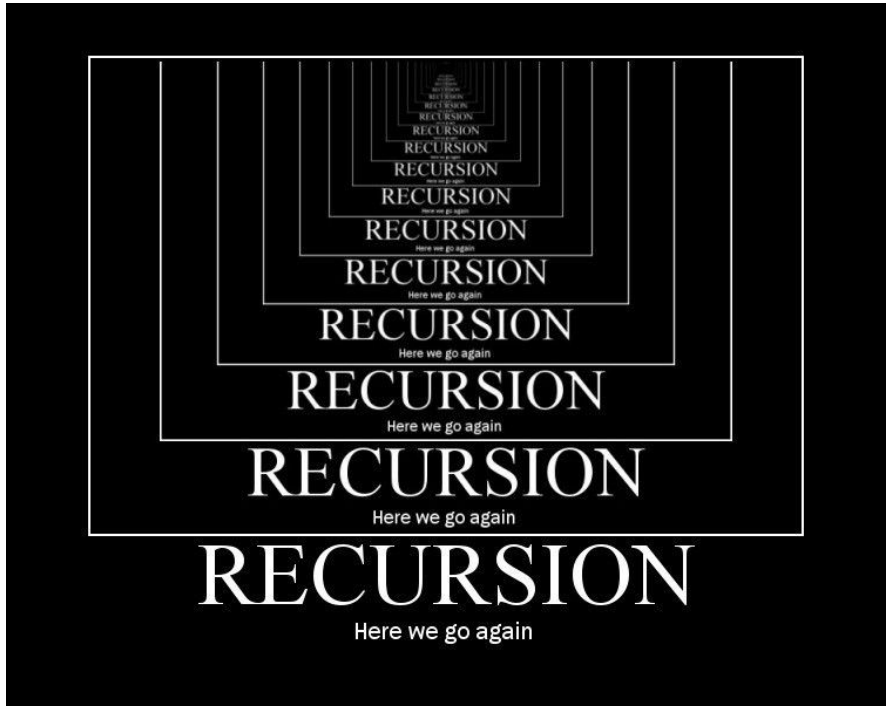
- Static = you don’t have to instantiate an object
- Non-static = you have to instantiate an object

```
StaticLibrary.method();

NonStaticObject a = new NonStaticObject();
a.method();
```

Recursion

- A function that calls itself
  - Must have a base case
  - Must have a reduction step

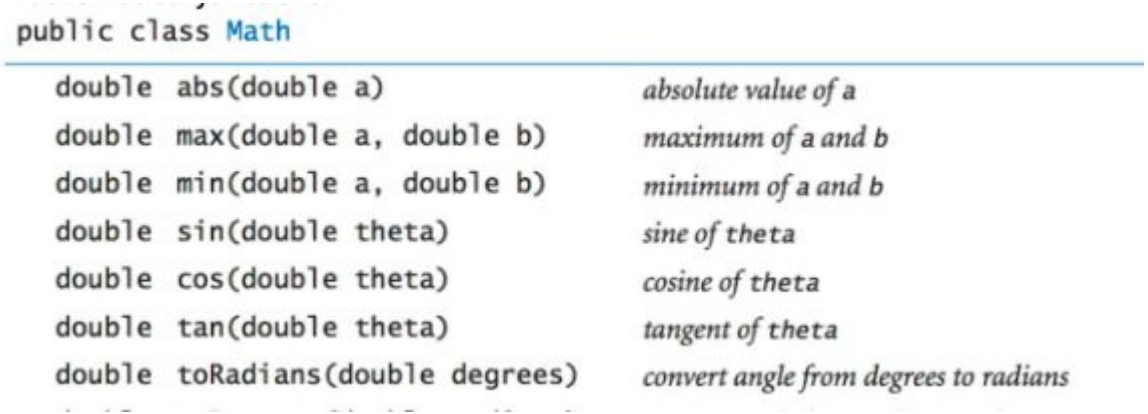


Libraries

- Don’t reinvent the wheel, use somebody else’s library!
- Contains **public** functions
  - Usually **static** as well
- Use libraries using **API**

API

- Application Programming Interface
- Contains information on:
  - Method name
  - Method parameters
  - Method return type



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*

Java Classes

- Libraries
  - Usually provide public functions
  - Usually **static** functions
  - e.g. Hyperbolic.java
- Data types
  - Usually are used as a template to make an object
  - Usually have a constructor
  - Usually non-**static** functions
  - e.g. Point.java

How do Java Classes relate to each other?

- Inheritance
- Interfacing

abstract keyword → no implementation

Used by interfaces and abstract classes

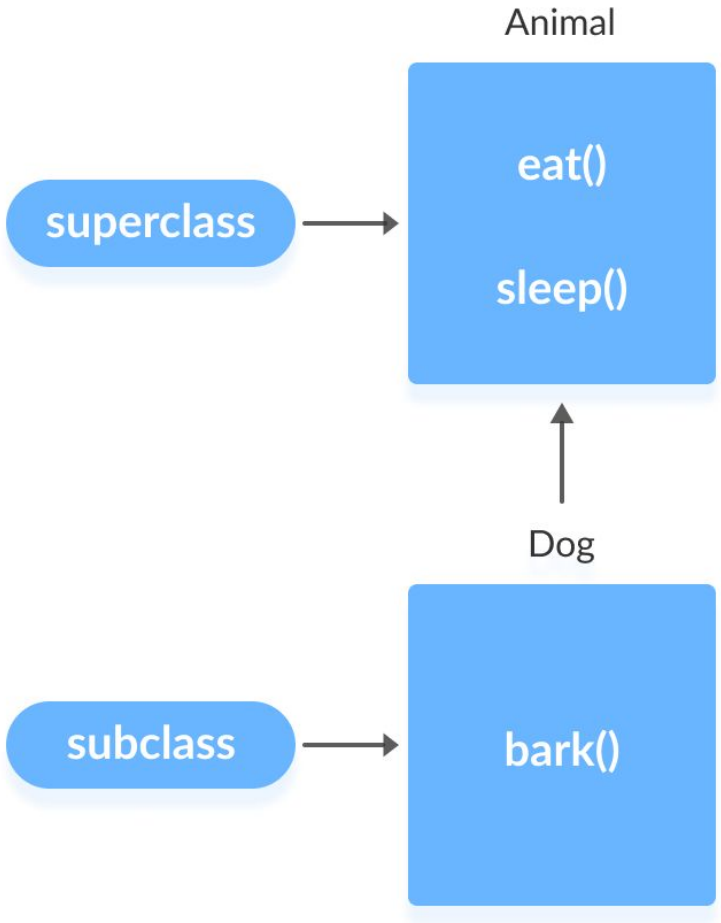
```
public abstract void abstractfunction();
```

Inheritance

```
public class Dog extends Animal {

    //.... code ...

}
```

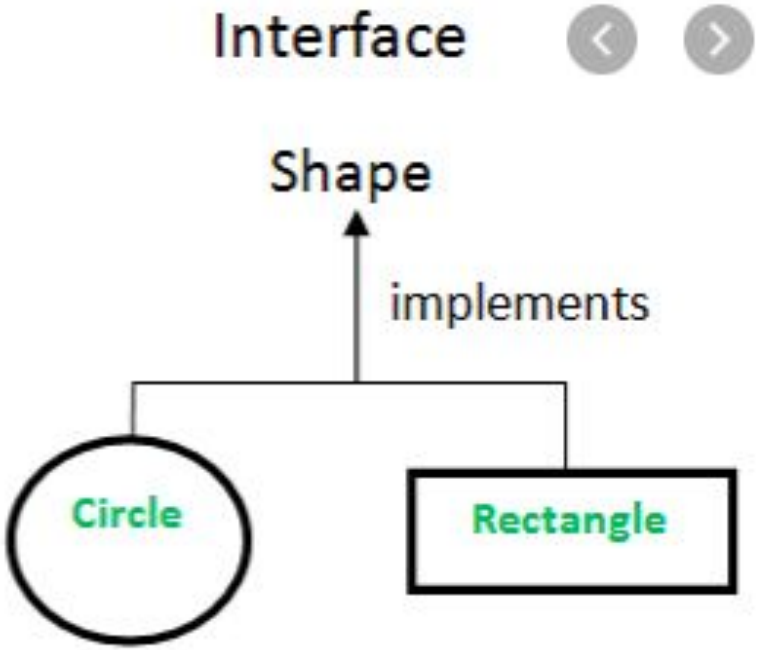


Interface

```
public class Rectangle implements Shape {

    //.... code ...

}
```



Some principles of good code design

- Encapsulation
- Immutability
- There are more! But we'll just cover these two for now

Encapsulation

- Have all the necessary information and functions together inside one class
- Hide state of an object from others by…
  - making variables private
- For example, let's say you have a variable named counter that should always be private. You can prevent stuff like this

```
Counter counter = new Counter("Volusia");
counter.count = -16022;
```

Encapsulation (cont'd)

Encapsulation:

```
counter.increment();
```

Not encapsulation:

```
notACounter.makeCounterIncrement();
// this class is changing state of another object
```

Immutability

- Prevent unwanted changes
  - Should others be changing the value of variables like PI?
- Immutability is preserved if...
  - Variables have the “final” keyword, OR
  - Variables private AND only the constructor accesses them
- String is an immutable object in Java ← common Java interview question

Is this class immutable?

```
import java.util.Date

public class Appointment {
    private Date date;
    private String contact;

    public Appointment(Date date) {
        this.date = date;
        this.contact = contact;
    }

    public Date getDate() {
        return date;
    }
```

Is this class immutable?

```
import java.util.Date

public class Appointment {
    private Date date;
    private String contact;

    public Appointment(Date date) {
        this.date = date;
        this.contact = contact;
    }

    public Date getDate() {
        return date;
    }
```

No, because Date might not be immutable!

Is this class immutable?

```
import java.util.Date

public class Appointment {
    private Date date;
    private String contact;

    public Appointment(Date date) {
        this.date = date;
        this.contact = contact;
    }

    public Date getDate() {
        return date;
    }
```

No, because Date might not be immutable!

But, if we know Date is immutable, then Appointment is also.