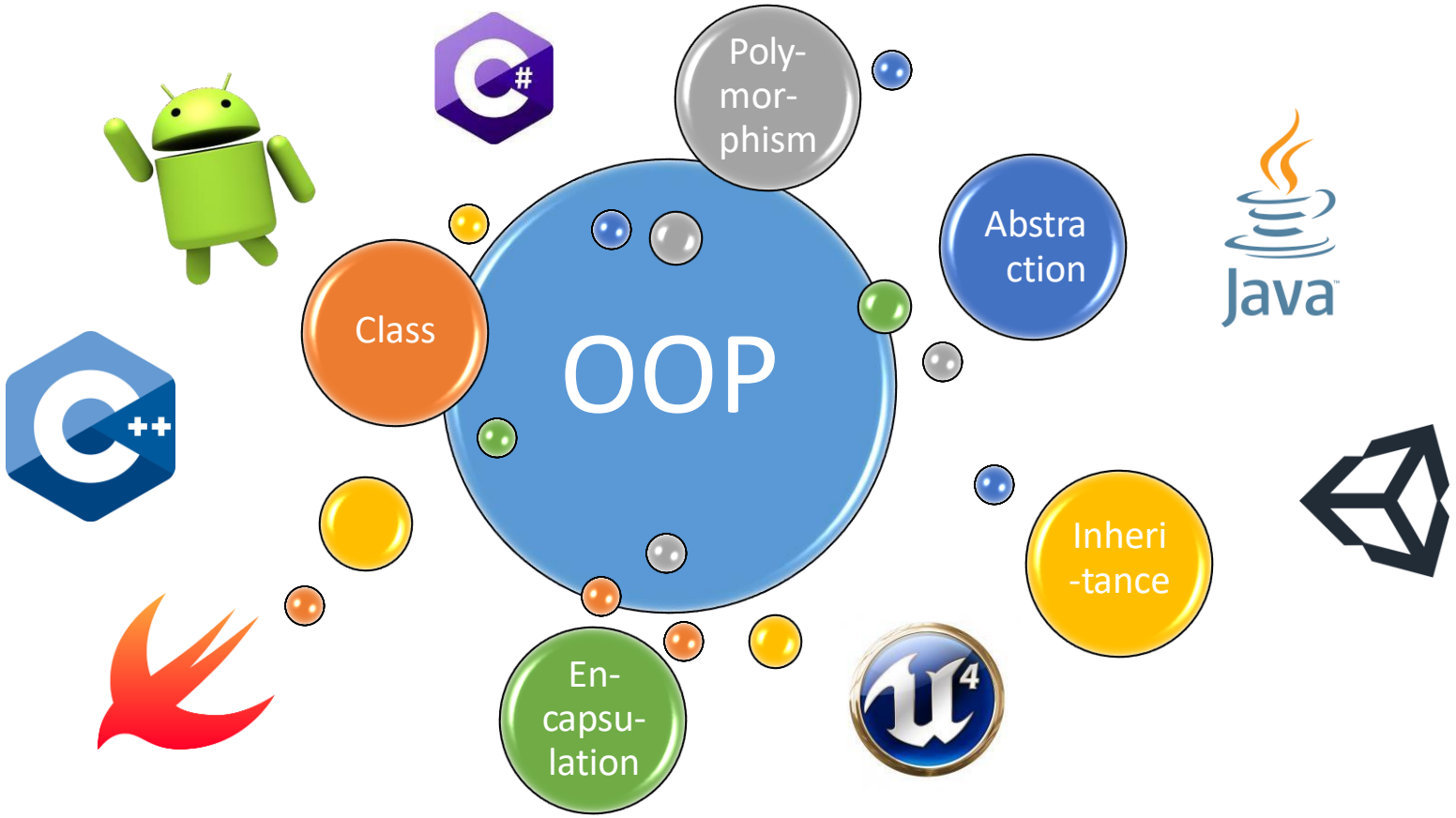
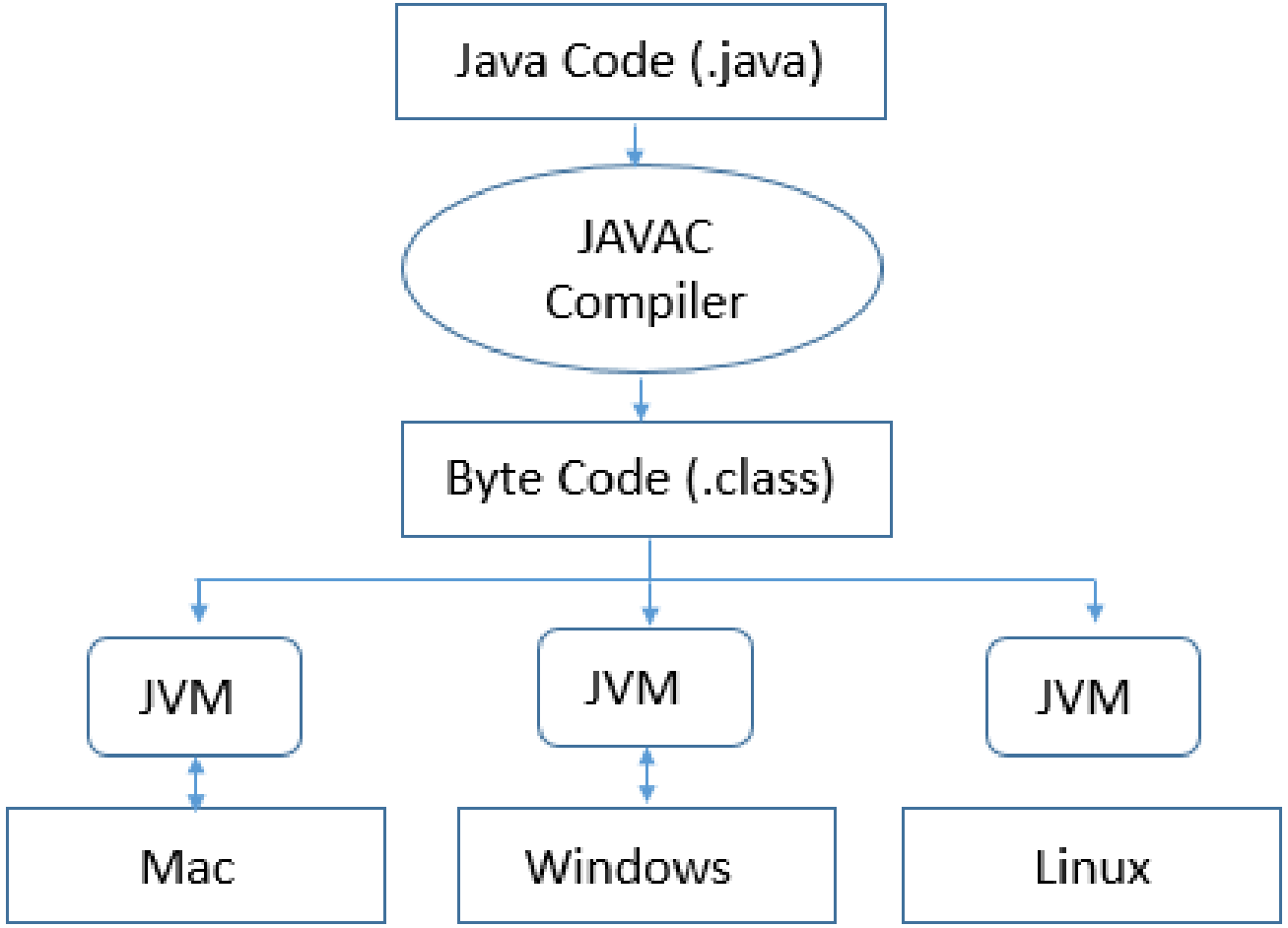


JAVABasics & OOP

Kuan-Ting Lai

Write Once, Run Anywhere

Java Virtual Machine



http://net-informations.com/java/intro/jvm.htm

Java Overview

- Simple
- Object-Oriented
- Robust & Secure
- Architecture-neutral and portable
- High Performance
- Interpreted, Threaded, and Dynamic

Install Java Development Kit (JDK)

- Download Java SE Development Kit from Oracle website
- https://www.oracle.com/java/technologies/javase-downloads.html#javasejdk

Java SE Downloads

Java Platform, Standard Edition

Java SE 14

Java SE 14.0.1 is the latest release for the Java SE Platform

- Documentation
- Installation Instructions
- Release Notes
- Oracle License
  - Binary License
  - Documentation License
- Java SE Licensing Information User Manual
  - Includes Third Party Licenses
- Certified System Configurations
- Readme

Oracle JDK

JDK Download

Documentation Download

Looking for Oracle OpenJDK builds?

- **Oracle Customers and ISVs targeting Oracle LTS releases:** Oracle JDK is Oracle's supported Java SE version for customers and for developing, testing, prototyping or demonstrating your Java

Install Java Development Kit (JDK)

- Download it and install it.

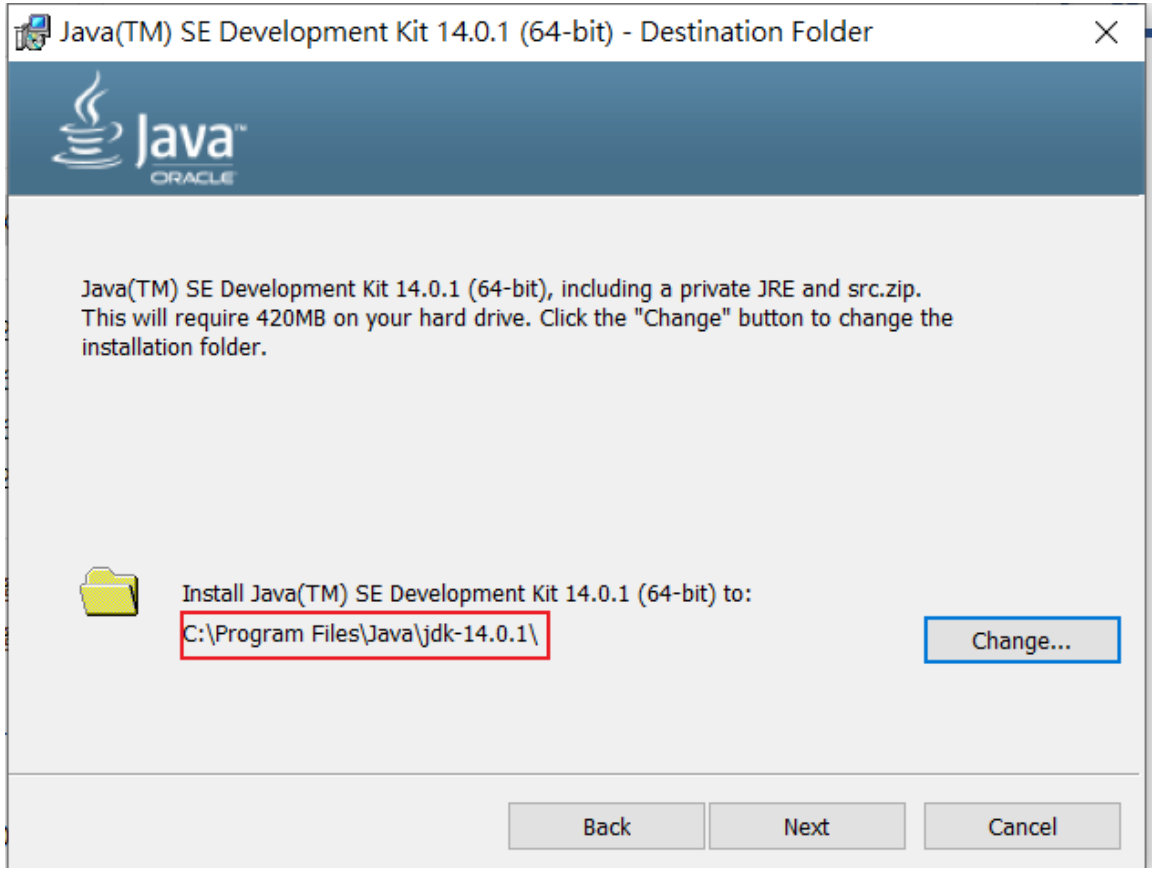
Java SE Development Kit 14

This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

Product / File Description	File Size	Download
Linux Debian Package	157.92 MB	jdk-14.0.1_linux-x64_bin.deb
Linux RPM Package	165.04 MB	jdk-14.0.1_linux-x64_bin.rpm
Linux Compressed Archive	182.04 MB	jdk-14.0.1_linux-x64_bin.tar.gz
macOS Installer	175.77 MB	jdk-14.0.1_osx-x64_bin.dmg
macOS Compressed Archive	176.19 MB	jdk-14.0.1_osx-x64_bin.tar.gz
Windows x64 Installer	162.07 MB	jdk-14.0.1_windows-x64_bin.exe
Windows x64 Compressed Archive	181.53 MB	jdk-14.0.1_windows-x64_bin.zip

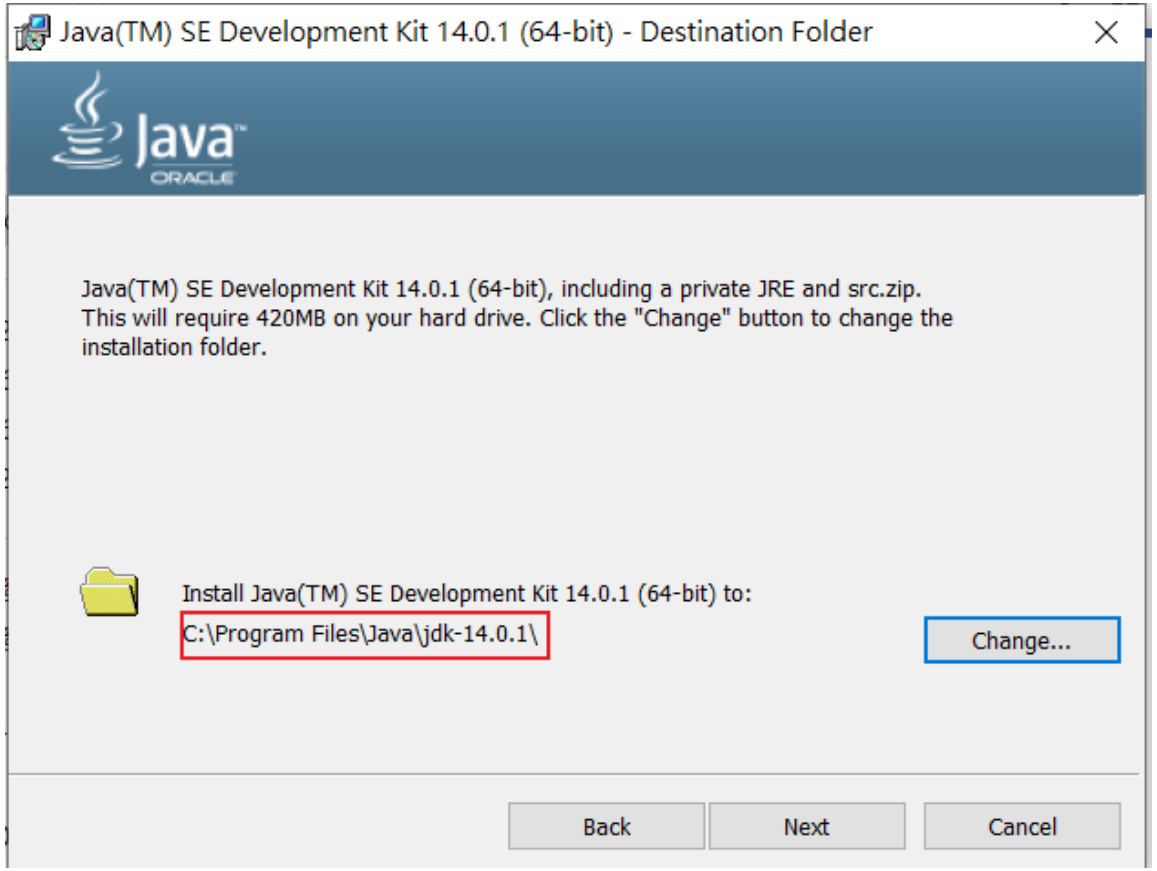
Install Java Development Kit (JDK)

- Download it and install it.
- Remember the path, we'll use it later.



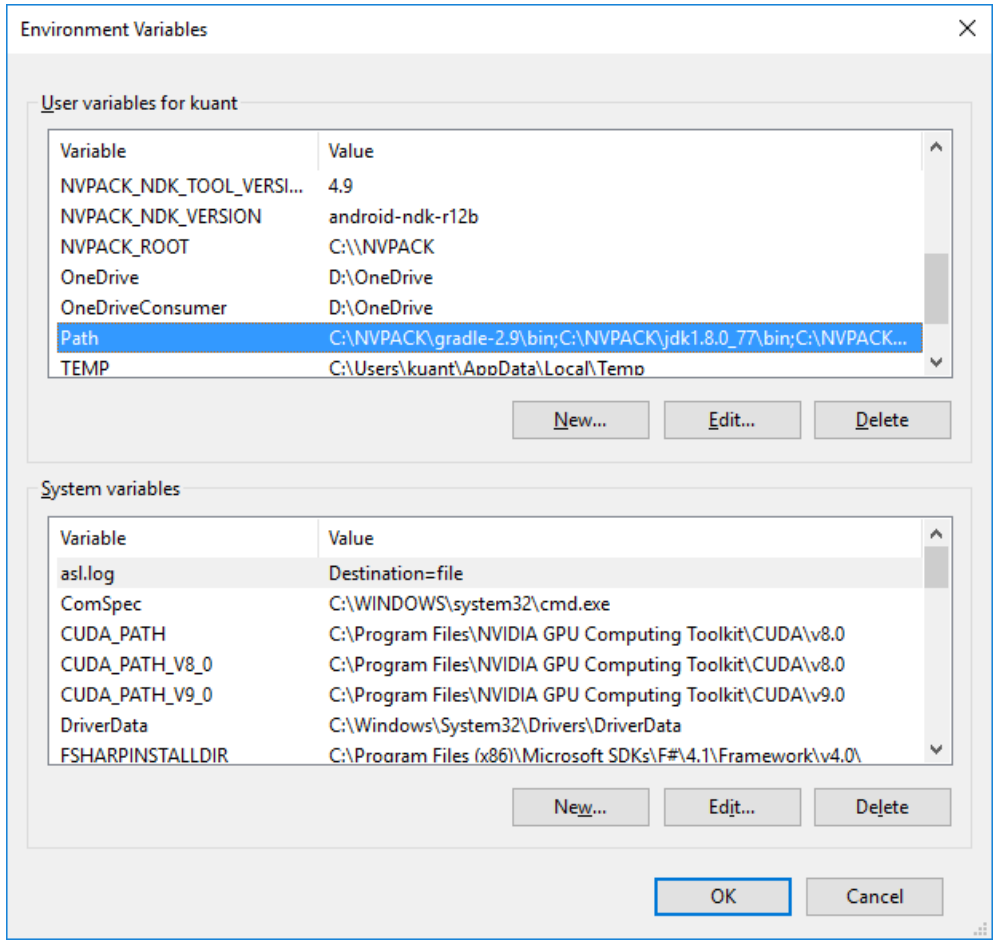
Install Java Development Kit (JDK)

- Download it and install it.
- Remember the path, we'll use it later.



Add JDKFolder to “Path” Variable

- Windows
  - 'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk-14.01\bin’
- Linux
  - export PATH = /path/to/java:$PATH



Test the tools

- Open the terminal and input:
- $ javac -version
- $ java -version

命令提示字元

```
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\chaoyu>javac -version
javac 14.0.1

C:\Users\chaoyu>java -version
java version "14.0.1" 2020-04-14
Java(TM) SE Runtime Environment (build 14.0.1+7)
Java HotSpot(TM) 64-Bit Server VM (build 14.0.1+7, mixed mode, sharing)

C:\Users\chaoyu>
```

First Java Program

- Compile and run first java program

  C:\> javac MyFirstJavaProgram.java C:\>

  java MyFirstJavaProgram

  Hello World

```
public class MyFirstJavaProgram {

    public static void main(String []args)
    {
        System.out.println("Hello World"); // prints Hello World
    }
}
```

Use Scanner

- C:\> javac ScannerDemo.java
- C:\> java ScannerDemo
- Please type your name:
- Hello! [your name]

```
import java.util.Scanner;

public class ScannerDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Please type your name：");
        System.out.printf("Hello！ %s!\n", scanner.next());
    }
}
```

Use BufferedReader

- C:\> javac BufferedReaderDemo.java
- C:\> java BufferedReaderDemo
- type anything:

```
import java.io.*;

public class BufferedReaderDemo {
    public static void main(String[] args) throws IOExc
eption {
        BufferedReader bufferedReader =
            new BufferedReader(new InputStreamReader(Sy
stem.in));

        System.out.print("type anything: ");
        String text = bufferedReader.readLine();
        System.out.println("your input: " + text);
    }
}
```

Basic Syntax

- **Class Names** – the first letter should be in Upper Case (CamelCase style)
  - Example: class MyFirstJavaClass
- **Method Names** – start with a Lower Case letter.
  - Example: public void myMethodName()
- **Program File Name** – should exactly match the class name.
  - *MyFirstJavaProgram.java*
- **public static void main(String args[])** – starting point, a mandatory part of every Java program.

Java Class

- Instance variables
  - width, height
- Class variables
  - total_shapes
- Local variables
  - w, h

```
public class Shape {
    int width;
    int height;
    static int total_shapes = 0;

    void setWidth(int w) {
        width = w;
    }

    void setHeight(int h) {
        height = h;
    }
}
```

Constructor

```
public class Shape {
    int width;
    int height;

    public Shape(int a, int b)        {
        width = a; height = b;
    }


     void setWidth(int w) {
         width = w;
     }
     void setHeight(int h) {
         height = h;
     }

}
```

Java Modifiers

- Access Control Modifiers
  - public, private, protected
- Non-Access Modifiers
  - static, final, abstract
  - synchronized, volatile

```
public class className {
    // ...
}
private boolean myFlag;
static final double weeks = 9.5;
protected static final int BOXWIDTH = 42;

public static void main(String[] arguments) {
    // body of method
}
```

Compile & Run Your First Java Program

- C:\> javac Shape.java
- C:\> javac ShapeTest.java
- C:\> java ShapeTest

  7.700000000000001

Shape.java

```
public class Shape {
    double width = 0;
    double height = 0;

    Shape(double a, double b) {
        width = a; height = b;
    }

    double area() {
        return width * height;
    }
};
```

ShapeTest.java

```
import java.io.*;

public class ShapeTest
{
    public static void main(String args[])
    {
        Shape shape = new Shape(3.5, 2.2);
        System.out.println(shape.area());
    }
}
```

Java Primitive Data Types

- byte (8-bit)
- short (16-bit)
- int (32-bit)
- long (64-bit)
- float (32-bit)
- double (64-bit)
- boolean (1-bit)
- char (16-bit Unicode character)

Java String

- Initialize a String

```
public class StringDemo {
    public static void main(String args[]) {
        char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.' };
        String helloString = new String(helloArray);
        System.out.println( helloString );
    }
}
```

- Concatenate String

```
public class StringDemo {
    public static void main(String args[]) {
        String string1 = "saw I was ";
        System.out.println("Dot " + string1 + "Tod");
    }
}
```

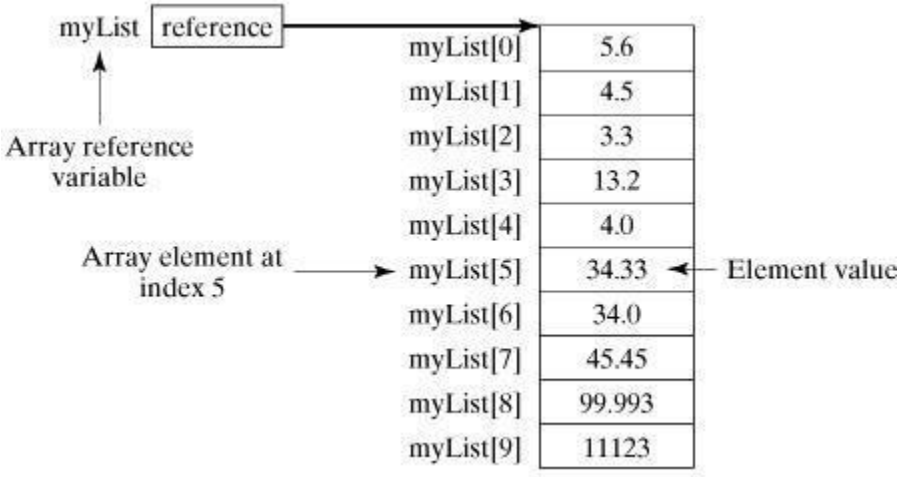
String Methods

Method Name	Description
int length()	Returns the length of this string.
char charAt(int index)	Returns the character at the specified index.
int compareTo(String anotherString)	Compares two strings lexicographically.
byte[] getBytes(String charsetName)	Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array.
int indexOf(int ch)	Returns the index within this string of the first occurrence of the specified character.
int lastIndexOf(int ch)	Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.
String replace(char oldChar, char newChar)	Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
String substring(int beginIndex)	Returns a new string that is a substring of this string.
String[] split(String regex)	Splits this string around matches of the given regular expression.
String trim()	Remove leading and trailing whitespace.

Java Array

- double[] myList = new double[10];

```
public class TestArray {
    public static void main(String[] args) {
        double[] myList = {1.9, 2.9, 3.4, 3.5};

        // Print all the array elements
        for (int i = 0; i < myList.length; i++) {
        System.out.println(myList[i] + " ");
    }
}
```



Java Date

- Date and Simple Date Format

```java
import java.util.Date;

public class DateDemo {
	public static void main(String args[]) {
		// Instantiate a Date object
		Date date = new Date();

		// display time and date using toString()
		System.out.println(date.toString());
	}
}
```

```
C:> on May 04 09:51:52 CDT 2009
```

```java
import java.util.*;
import java.text.*;

public class DateDemo {
	public static void main(String args[]) {
		Date dNow = new Date( );
		SimpleDateFormat ft =
		new SimpleDateFormat ("E yyyy.MM.dd 'at' hh:mm:ss a zzz");
		System.out.println("Current Date: " + ft.format(dNow));
	}
}
```

```
C:> Current Date: Sun 2004.07.18 at 04:14:09 PM PDT
```

Java Operators

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Decision Making

- if statement

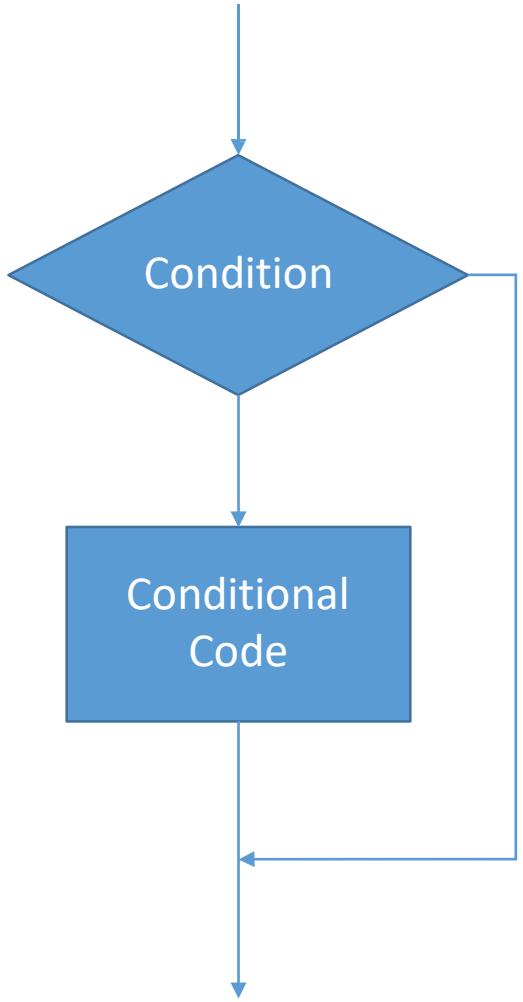
```
if (....) {
    ...
} else if (....) {
    ...
} else {
    ...
}
```

- switch statement

```
switch (....) {
    case 0: ...
        break;
    case 0: ...
        break;
    default: ...
}
```

- ? : operator

```
Exp1 ? Exp2 : Exp3;
```



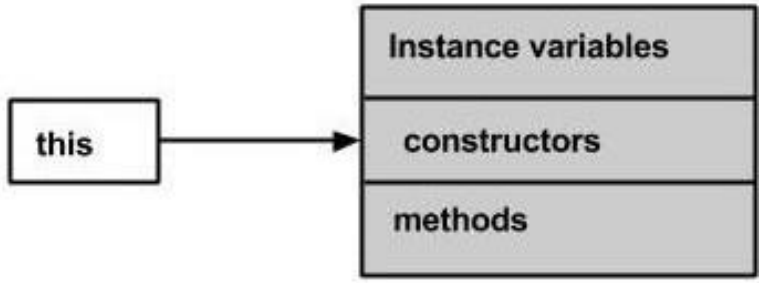
Loop Control Statements

- while
- for
- do while
- for(declaration : expression)

```
public class Test {
        public static void main(String args[]) {
                int [] numbers = {10, 20, 30, 40, 50};

                for(int x : numbers ) {
                    System.out.print( x );
                    System.out.print(",");
                }
                System.out.print("\n");
                String [] names = {"James", "Larry", "Tom", "Lacy"};

                for( String name : names ) {
                    System.out.print( name );
                    System.out.print(",");
                }
        }
}
```

“this” pointer

- **this** is a keyword in Java which is used as a reference to the object of the current class, with in an instance method or a constructor

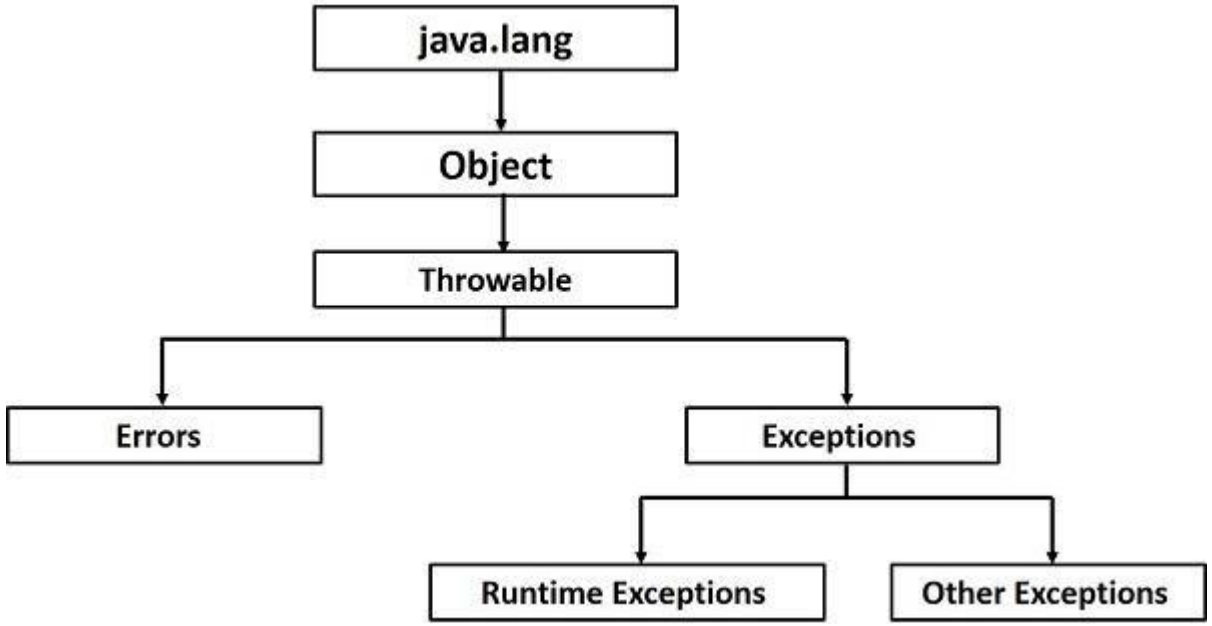



```
class Student {
    int age Student() {
        this(20);
    }

    Student(int age) {
        this.age = age;
    }
}
```

Handling Exceptions

- Errors are abnormal conditions that happen in case of severe failures, these are not handled by the Java programs
  - Example: JVM is out of memory
- Two main subclasses: IOException class and RuntimeException Class

```
try {
    // Protected code
}
catch (ExceptionName e1) {
    // Catch block
}
```



Catching Multiple Exceptions

```
try {
    file = new FileInputStream(fileName);
    x = (byte)file.read();
}
catch (IOException i) {
    i.printStackTrace();
    return -1;
}
catch (FileNotFoundException f) {
    f.printStackTrace();
    return -1;
}
finally {
    System.out.println("The finally statement is executed");
}
```

Java Sleep

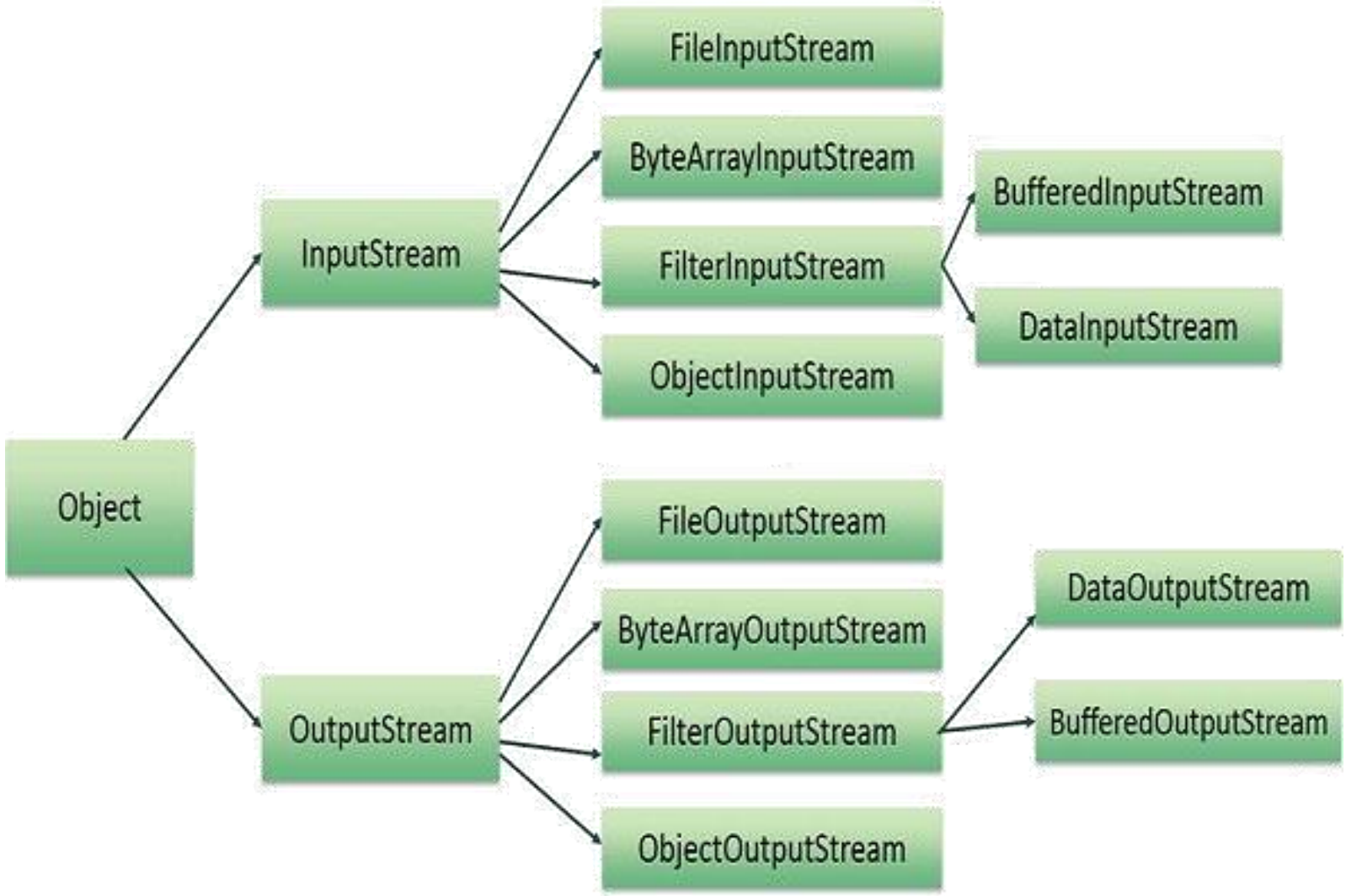
- Thread.sleep()

```
import java.util.*;

public class SleepDemo {
    public static void main(String args[]) {
        try {
            System.out.println(new Date( ) + "\n");
            Thread.sleep(5*60*10);
            System.out.println(new Date( ) + "\n");
        }
        catch (Exception e)
        {
            System.out.println("Got an exception!");
        }
    }
}
```

Java Files and I/O

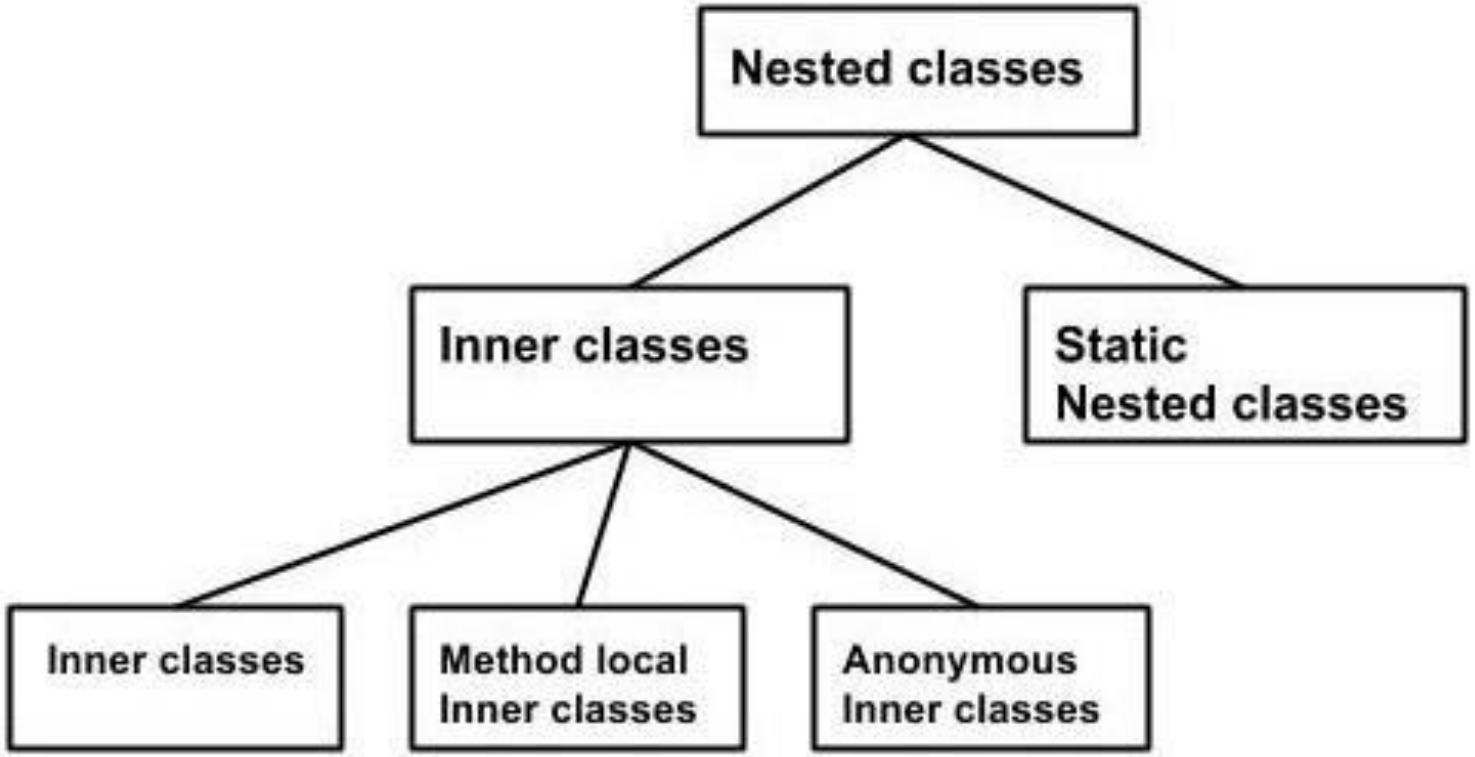
```
import java.io.*;
public class CopyFile {
    public static void main(String args[]) throws IOException {
        FileInputStream in =null;
        FileOutputStream out =null;
        try {
            in =new FileInputStream("input.txt");
            out =new FileOutputStream("output.txt");
            int c;
            while ((c =in.read()) != -1)
                out.write(c);
        }
        finally {
            if (in != null)
                in.close();
            if (out !=null)
                out.close();
        }
    }
}
```

Java IO Inheritance Hierarchy



Inner Class

- Writing a class within another



Inheritance

- **extends** is the keyword used to inherit the properties of a class

```
class SuperClass {
.....
.....
}
class SubClass extends SuperClass {
.....
.....
}
```

Keyword “super”

- Access parent (super) class

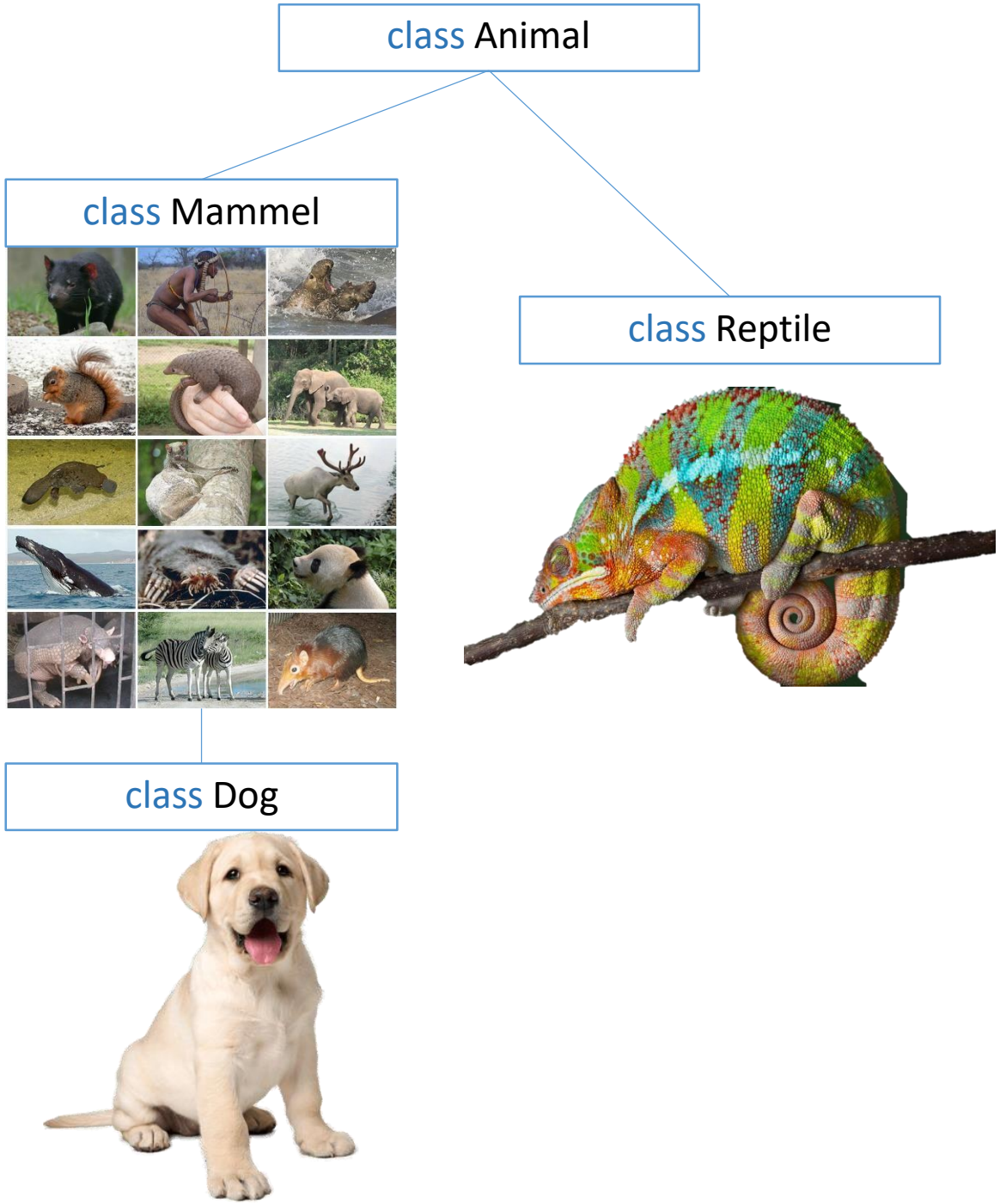
```
class Superclass {
    int age;
    Superclass(int age) {
        this.age = age;
    }
    public void getAge() {
        System.out.println("The value of the variable namedage in super class is: " + age);
    }
}
public class Subclass extends Superclass {
    Subclass(int age) {
        super(age);
    }
    public static void main(String args[]) {
        Subclass s = new Subclass(24);
        s.getAge();
    }
}
```

https://stackoverflow.com/questions/180601/using-super-in-c

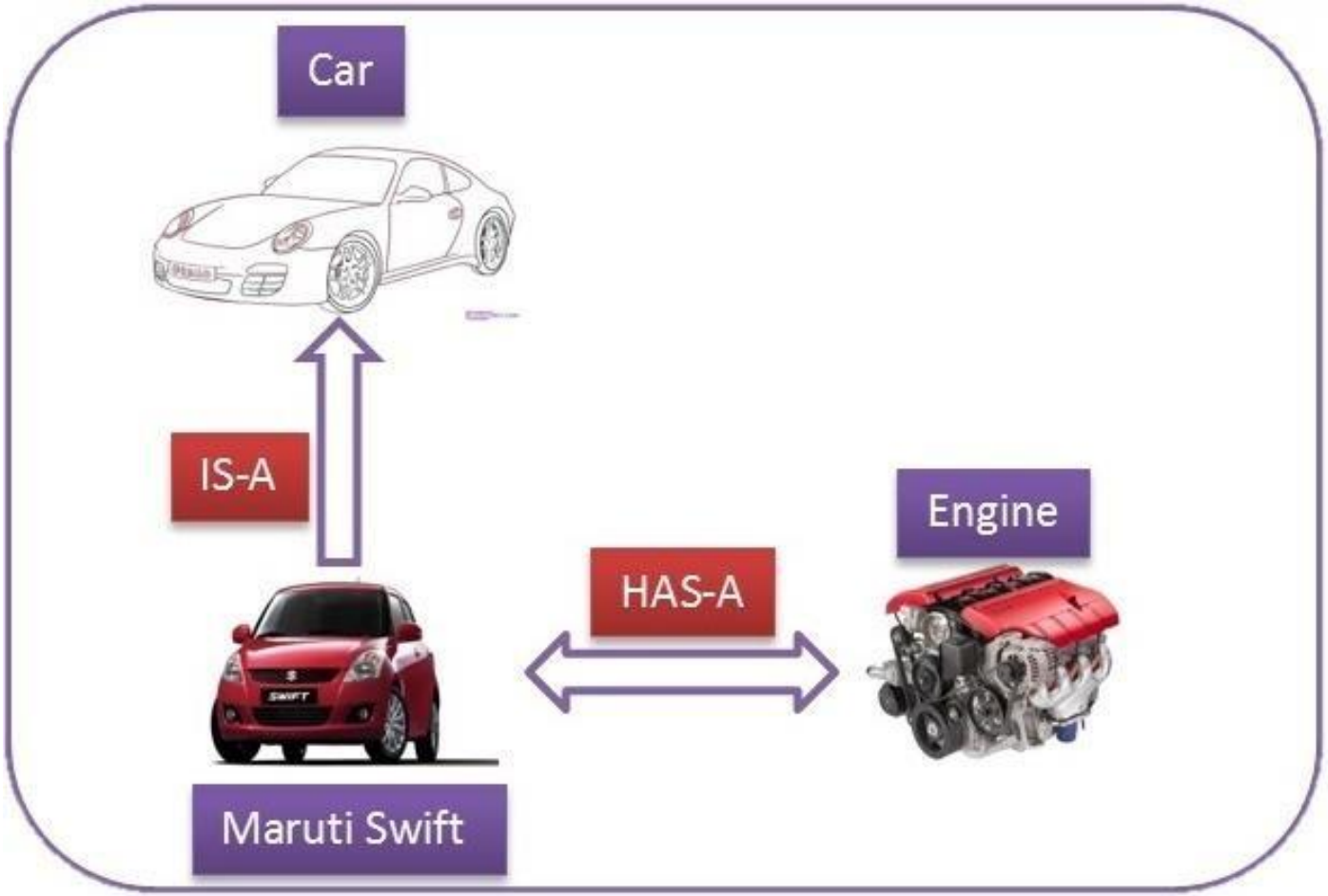
IS-A Relationship

- Mammal IS-A Animal
- Reptile IS-A Animal
- Dog IS-A Mammal
- Hence: Dog IS-A Animal as well

```
public class Animal {
}

public class Mammalextends Animal {
}

public class Reptile extends Animal{
}

public class Dog extends Mammal {
}
```



HAS-A Relationship



https://ramj2ee.blogspot.com/2015/08/java-tutorial-inheritance-has.html

Keyword “instanceof”

```
class Animal {}
class  Mammal implements  Animal {}

public class  Dog  extends  Mammal {

    public static void main(String args[]) {
        Mammal m = new Mammal();
        Dog d = new Dog();

        System.out.println(m  instanceof  Animal);
        System.out.println(d  instanceof  Mammal);
        System.out.println(d instanceof Animal);
    }
}
```

Types of Inheritance

Single Inheritance
Class A
↑
Class B
	public class A {
	.....
	}
	public class B **extends** A {
	.....
	}

Multi Level Inheritance Class A ↑ Class B ↑ Class C	public class A { .....} public class B **extends** A {.....} public class C **extends** B {..... }
Hierarchical Inheritance Class A ↑ ↑ Class B Class C	public class A { .....} public class B **extends** A {.....} public class C **extends** A {..... }

Multiple Inheritance
Class A Class B
↖ ↗
Class C
	public class A { .....}
	public class B {.....}
	public class C **extends** A,B {
	.....
	} // Java does not support mutiple Inheritance

Overriding

- Overriding can be prevented by using modifier “final”

```
class Animal {
    public void move() {
        System.out.println("Animals can move");
    }
}
class Dog extends Animal {
    public void move() {
        System.out.println("Dogs can walk and run");
    }
}
public class TestDog {
    public static void main(String args[]) {
        Animal a = new Animal();    // Animal reference and object
        Animal b = new Dog();    // Animal reference but Dog object

        a.move();    // runs the method in Animal class
        b.move();    // runs the method in Dog class
    }
}
```

Java Abstraction

- Define the functionality but hide the details of implementation
- Abstract Class
  - if a class has at least one abstract method, then the class **must** be abstract
  - If a class is declared abstract, it cannot be instantiated
  - To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
- Abstract Method

```
public abstract class Shape {
    void setWidth(int w) {
        width = w;
    }
    void setHeight(int h) {
        height = h;
    }
    public abstract int  area();

    int width;
    int height;
};
```

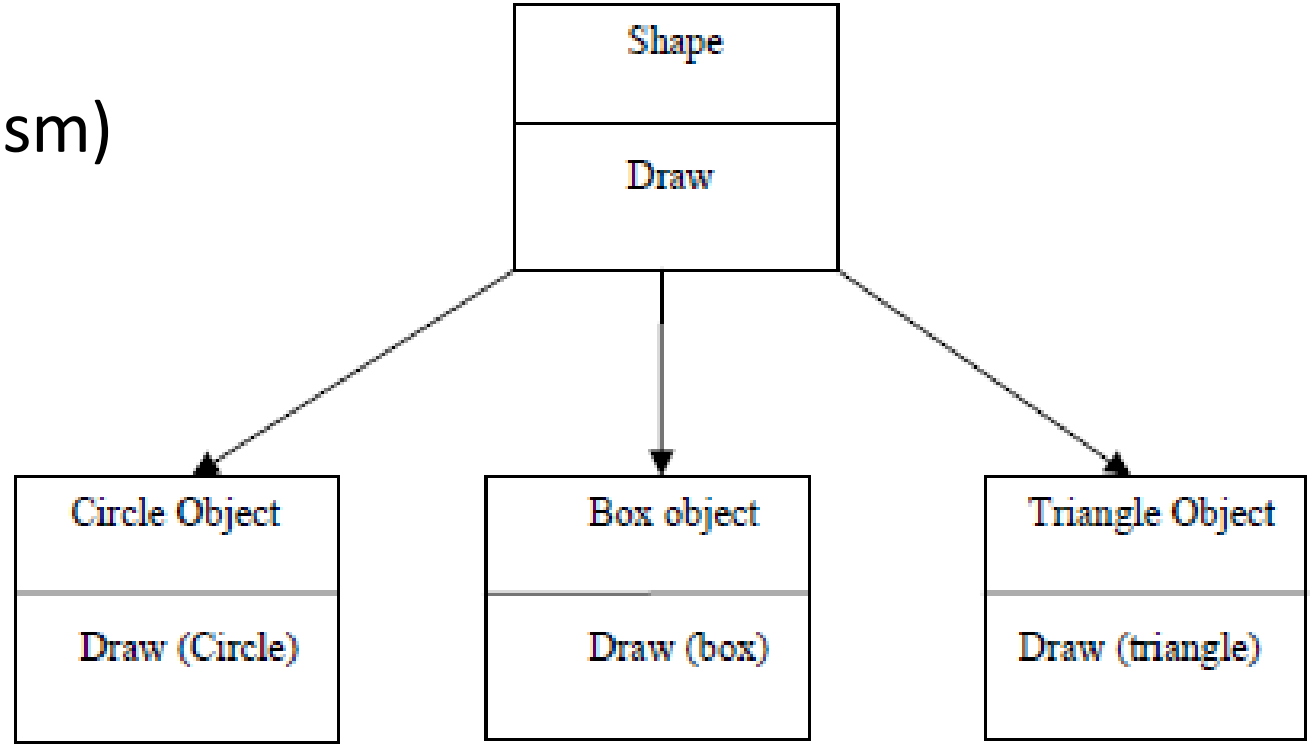
Java Encapsulation

- Declare the variables of a class as private.
- Provide public setter and getter methods to modify and view the variables values.

```java
class Student {
    private String name;
    private String id;
    private int age;

    public int getAge() {
        return age;
    }
    public String getName() {
        return name;
    }
    public String getId() {
        return id;
    }
    public void setAge(int newAge) {
        age = newAge;
    }
    public void setName(String newName) {
        name = newName;
    }
    public void setId(String newId) {
        id = newId;
    }
}
```

Polymorphism

- Overriding (Dynamic Polymorphism)
- Overloading (Static Polymorphism)



Java Interface

- Interface cannot be inherited
- Interface is implcitly abstract
- All of the methods in an interface are abstract
- An interface cannot contain instance fields, only static and final field
- Interface can extend multiple interfaces

```
interface Animal {   public
    void eat();   public
    void travel();
}

public class  MammalInt implements  Animal {
    public void eat() {
        System.out.println("Mammal eats");
    }
    public void travel() {
        System.out.println("Mammal travels");
    }
    public static void main(String args[]) {
        MammalInt m = new MammalInt();
        m.eat();
        m.travel();
    }
}
```

Java Interface

- Extending Multiple Interfaces

```
public interface Hockey extends Sports, Event
```

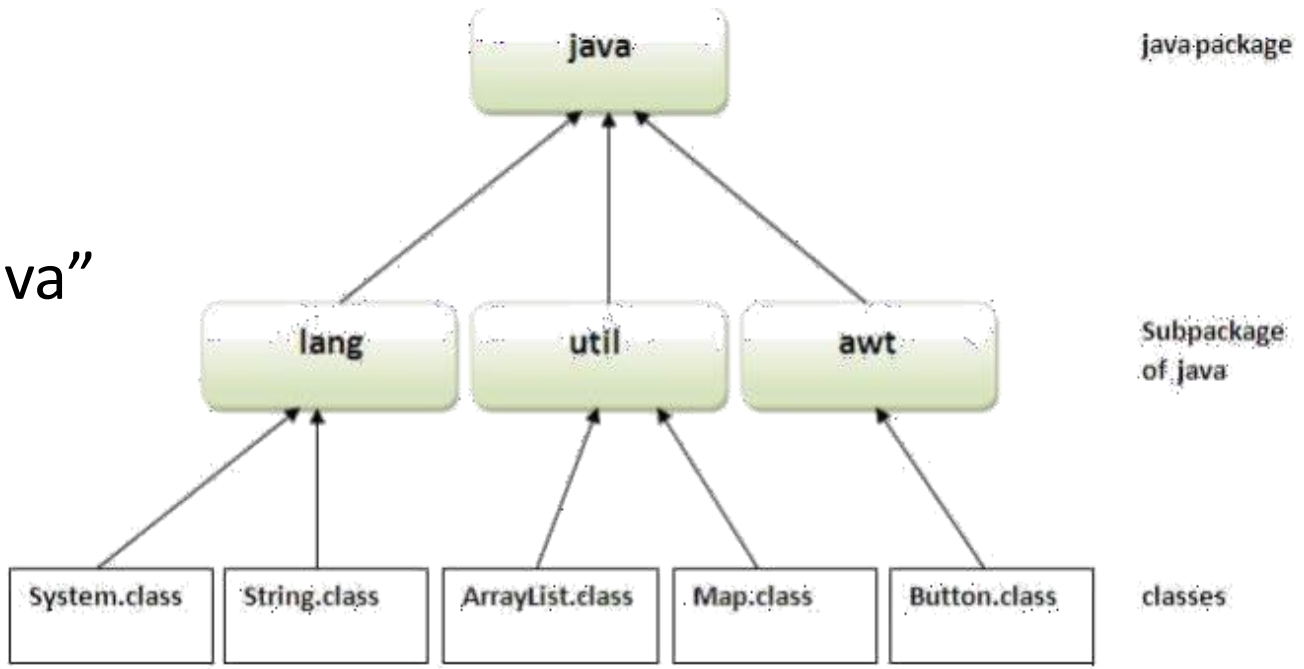
- Tagging Interfaces

```
package java.util;
public interface EventListener {}
```

Java Packages

- Prevent name conflicts
- Each package maps to a folder
  - .\com\apple\computers\Dell.java
- Use keyword “package” in “Dell.java”

```
package com.apple.computers;
public class Dell {
...
}
...
```



- Compile a package
  - javac -d destination_folder file_name.java
- Useage
  - import java.lang.String
  - import java.lang.*

UTF-8 problem

- $ javac -encoding "UTF-8" XXX.java
- $ java -Dfile.encoding="UTF-8" XXX

References

- https://www.tutorialspoint.com/java/
- What is Java Virtual Machine?
- https://www.softwaretestingmaterial.com/operators-in-java/