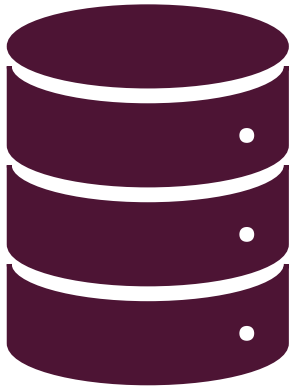


DATA STRUCTURES

WENYE LI

CUHK-SZ

ABOUT ME

Systematically trained in CS

- Familiarity with ~20 programming languages, including Java, C, Matlab, …
- Started to use Java from 1998, developed several systems still being used

Expectation/philosophy/style about the course:

- Target at training future leaders in related areas
- Illustrate complicated techniques in simple and plain words
- Same requirement for both CS and non-CS students
- Tough in coursework, generous in GPA

CONTACT & TA

Contact

- Rm. 413, Daoyuan Building
- wyli@cuhk.edu.cn
- Tuesday 14:00—15:00
- Thursday 17:00—18:00

TAs:

- Miss Mickey Ma, Mr. Xingjian Wang, Mr. Jianjun Wu, Mr. Yueyao Yu
- USTFs on the way, after add/drop period

ABOUT THIS COURSE

Lecture: 3 hours per week

Fundamental knowledge

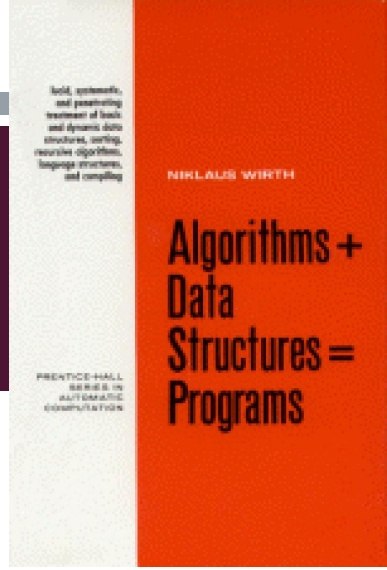
Tutorial: 1 hour per week

Skill improvement with leetcode examples (https://leetcode.com/)

Assessment:

4 Java programming projects (40%)
Midterm test (20%)
Final exam (40%)

ABOUT THIS COURSE



- One of the most important courses in CS
- Programs = Data Structures + Algorithms
  - Niklaus Wirth (Turing Award 1984)

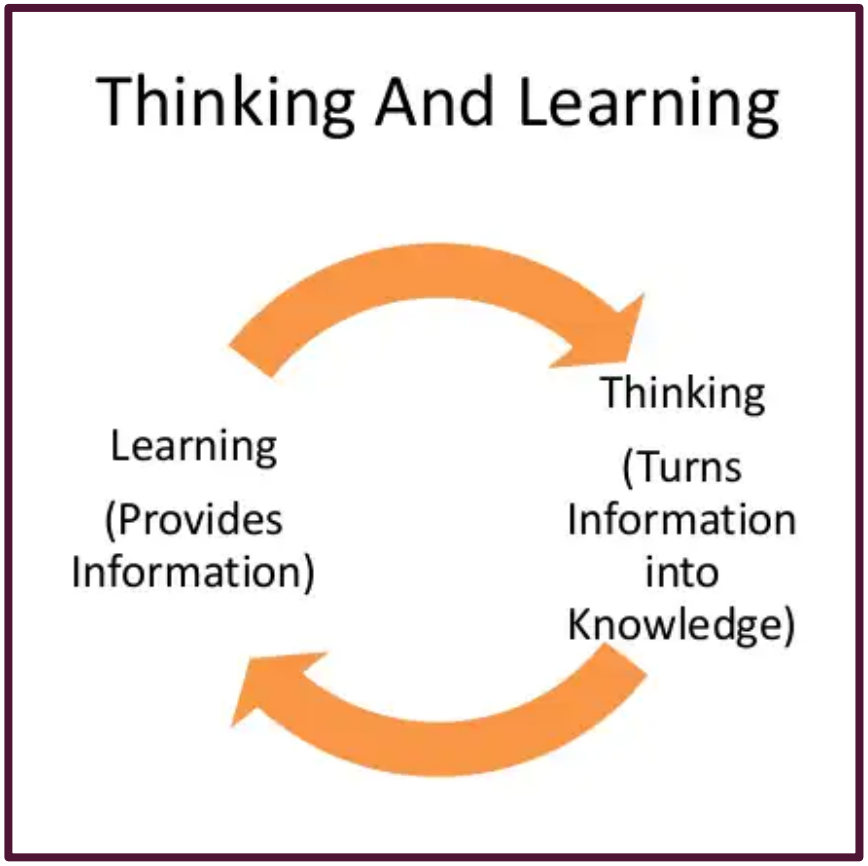
- A data structure is a data organization, management, and storage format that enables efficient access and modification.
- More precisely, a data structure is:
  - a collection of data values
  - the relationships among them
  - the operations that can be applied to the data

ABOUT THIS COURSE

- Relationship with other courses
  - CSC1001 (Python): focus on an entry-level programming language
  - CSC3002 (Programming Paradigm): use C/C++ to illustrate advanced language features
  - CSC4120 (Algorithms): design and analysis of advanced computer algorithms
  - This course: detailed illustration of data structures, with basic operations (algorithms) that can be performed on the structures

  - Overlapped knowledge with other courses is NOT this course's fault.
    - Fundamental knowledge is used everywhere. I use primary math every day, but not calculus.

HOW TO LEARN WELL?



- Thinking and Learning
  - Teacher: make a thick book into plain illustration
  - Student: organize simple ideas into deep knowledge
- Practice makes perfection
  - Expected to finish 100,000 lines of code in 4-year CS study.
  - How about you?

CHOOSE THIS COURSE OR NOT?

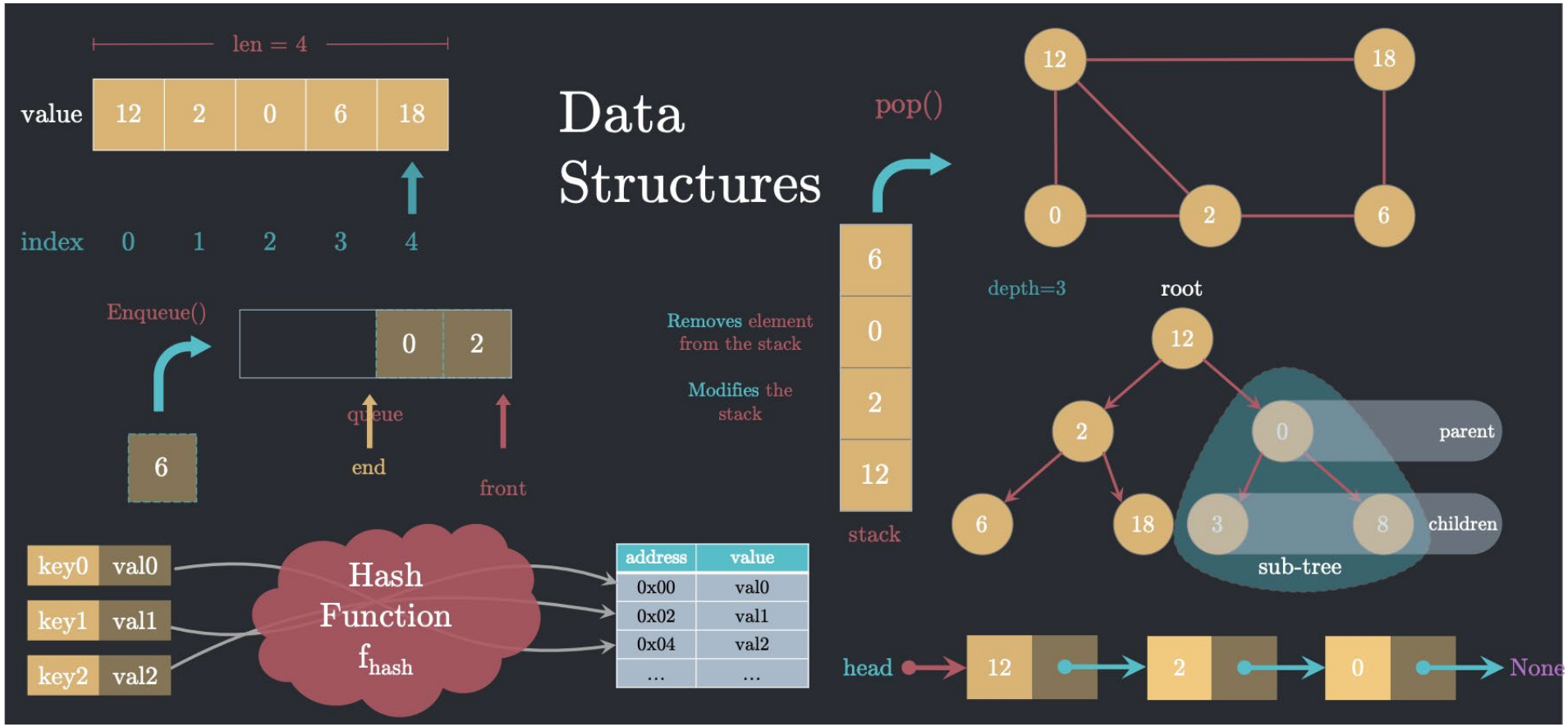
- This course has both theory and practice: 50% vs 50%
- Not many prerequisites, CSC1001 only. Fine for both CS and non-CS students

- However, not all like heavy load of programming.
  - Make decision as early as possible.
  - Quit does not imply being not good enough. It just means not your cup of tea.

COURSE OBJECTIVES

Know	the importance of data structure
Grasp	concepts/implementations of data structures
Understand	advantages/disadvantages of each data structure
Apply	key and fundamental algorithms
Analyze	basic algorithms' complexity and correctness
Get	prepared for further advanced studies

COURSE IN ONE FIGURE

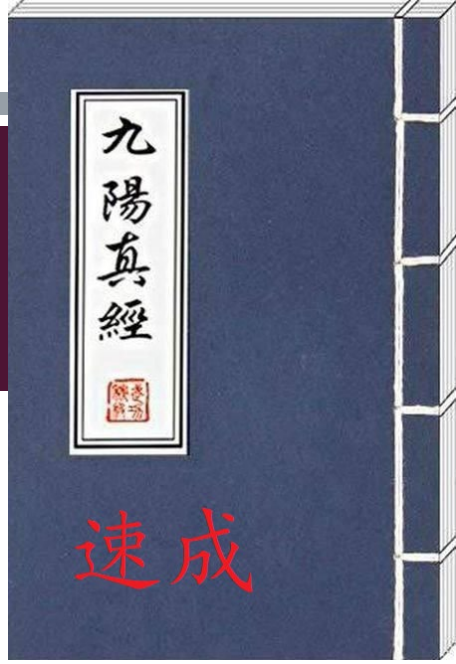


If you have understood all figure, don't waste time in classroom. Only come back for final exam.

COURSE PLAN

- On average, ~1.5 weeks for each topic:
  - Java Language Essentials (Project 1)
  - Arrays
  - Lists
  - Stacks and Queues (Project 2)
  - Trees
  - Midterm Test
  - Graphs (Project 3)
  - Other Data Structures
  - Sorting and Searching (Project 4)
  - Final Exam

JAVA ESSENTIALS



- We learn CSC1001 (Python) in a semester.
- After the 1st language, we only have 1~2 weeks to learn a 2nd.
- How can we learn a programming language quickly?
  - Examples + Pressure
  - You never know your potential without try. Trust me, you can.
- This is NOT a language course.
  - Learn grammar by yourself, not in classroom.

SETUP JAVA ENVIRONMENT

- Install a JDK (Java Development Kit).
  - Install an OpenJDK distribution.
  - Ensure that the following environment variables are set:
    - JAVA_HOME: Points to the base of the JDK installation.
    - PATH: Includes $JAVA_HOME/bin.
- Install/Choose an IDE or editor.
  - Notepad, Sublime Text Editor, Eclipse, VS Code …
- Test the first program
- More details: https://www.wikihow.com/Set-Up-a-Java-Programming-Environment

JAVA ESSENTIALS

- Java is a **programming language** and a **platform**, developed by *Sun Microsystems* in 1995.
  - Java is a high level, robust, object-oriented and secure programming language.
  - Java has a runtime environment (JRE) and API, called a platform.
- Java Applications
  - Desktop Applications such as acrobat reader, media player, antivirus, etc.
  - Web Applications such as websites
  - Enterprise Applications such as banking applications.
  - Mobile
  - Embedded System
  - Smart Card
  - Robotics
  - Games, etc.

JAVA ESSENTIALS: LANGUAGE FEATURES

- Simple
- Object-Oriented
- Portable
- Platform independent
- Secured
- Robust
- Architecture neutral
- Interpreted
- High Performance
- Multithreaded
- Distributed
- Dynamic

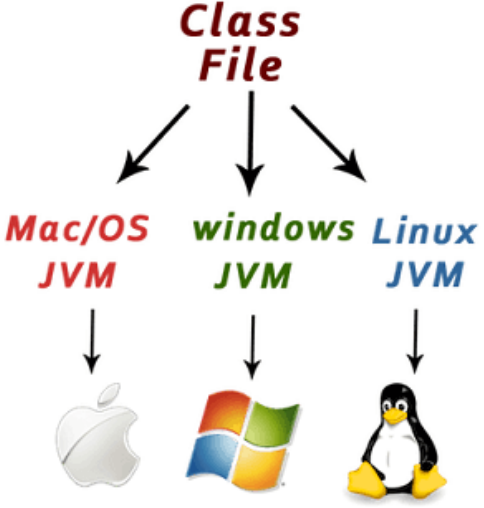
https://www.javatpoint.com/features-of-java

JAVA ESSENTIALS: OBJECT-ORIENTED

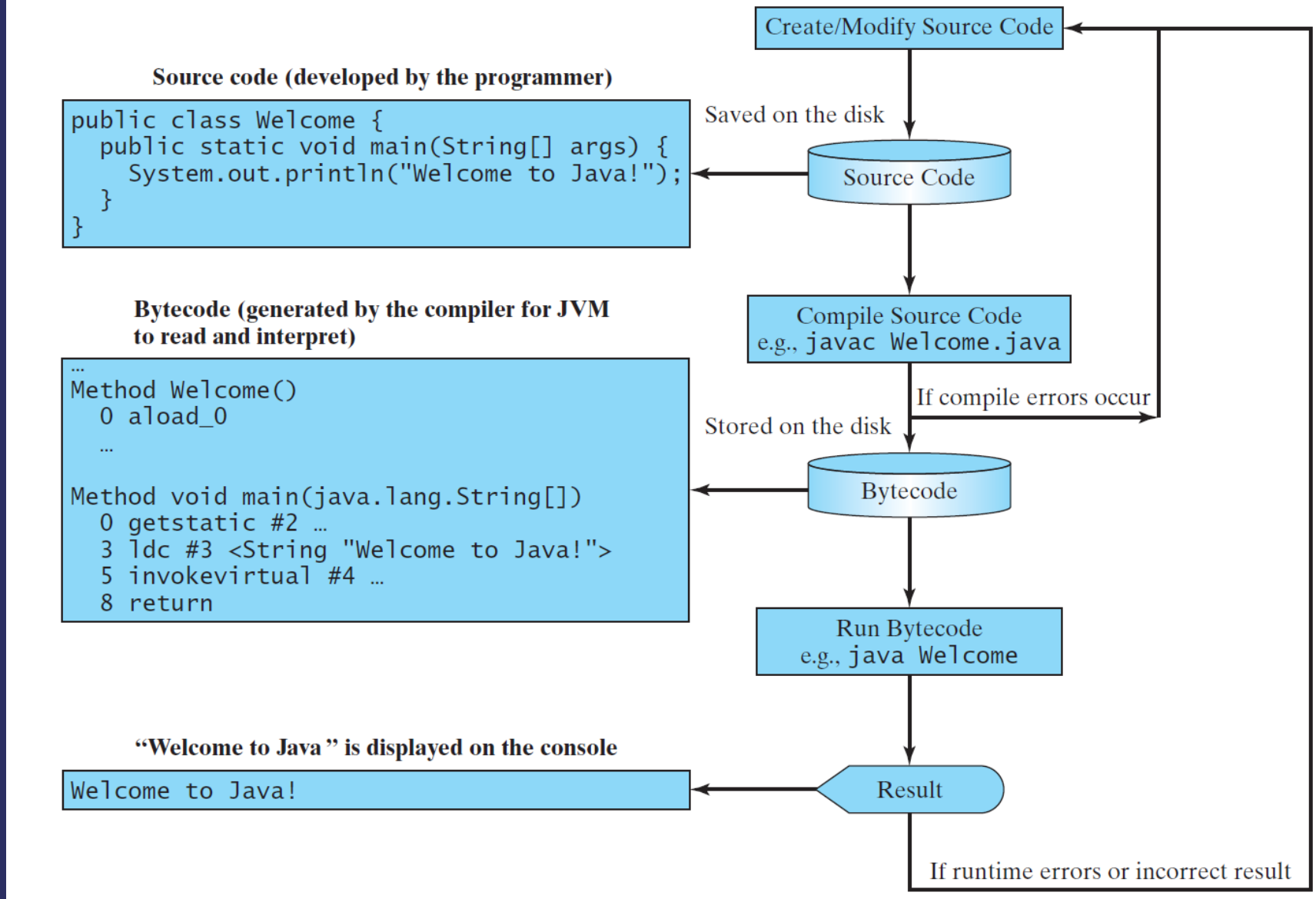
- Everything in Java is an object.
  - Software: a combination of different types of objects that incorporate both data and behavior.

- Object-oriented programming (OOPs) features:
  - Object
  - Class
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation

JAVA ESSENTIALS: PLATFORM INDEPENDENT



- Java: Write Once, Run Anywhere
  - Different from C/C++/…, which are compiled into platform specific machines.
- Java is a software platform that runs on top of hardware platforms.
  - Runtime Environment
  - API (Application Programming Interface)
- Java code can be executed on Windows, Linux, Mac/OS, etc.
  - Java program is compiled into bytecode.
  - The bytecode runs on multiple platforms.


Create/Modify Source Code
Source code (developed by the programmer)
public class Welcome {
public static void main(String[] args) {
System.out.println("Welcome to Java!");
}
}
Saved on the disk
Source Code
Compile Source Code
e.g., javac Welcome.java
If compile errors occur
Bytecode (generated by the compiler for JVM to read and interpret)
...
Method Welcome()
0 aload_0
...
Method void main(java.lang.String[])
0 getstatic #2 ...
3 ldc #3 <String "Welcome to Java!">
5 invokevirtual #4 ...
8 return
Stored on the disk
Bytecode
Run Bytecode
e.g., java Welcome
"Welcome to Java" is displayed on the console
Welcome to Java!
Result
If runtime errors or incorrect result

EXAMPLE: FIBONACCI SERIES

- In Fibonacci series, next number is the sum of previous two numbers.
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 etc.
  - The first two numbers of Fibonacci series are 0 and 1.

- Two ways to write the Fibonacci series program:
  - Fibonacci Series without using recursion
  - Fibonacci Series using recursion

```java
class FibonacciExample1
{
    public static void main(String args[])
    {
        int n1 = 0, n2 = 1, n3, i, count = 10;
        System.out.print(n1 + " " + n2); //printing 0 and 1

        for(i = 2; i < count; ++i) //loop starts from 2 because 0 and 1 are already printed
        {
            n3 = n1 + n2;
            System.out.print(" " + n3);
            n1 = n2;
            n2 = n3;
        }

    }
}
```

https://compiler.javatpoint.com/opr/test.jsp?filename=FibonacciExample1

```
0 1 1 2 3 5 8 13 21 34
```

```java
class FibonacciExample2
{
    static int n1 = 0, n2 = 1, n3 = 0;
    static void printFibonacci(int count)
    {
        if(count > 0)
        {
            n3 = n1 + n2;
            n1 = n2;
            n2 = n3;
            System.out.print(" " + n3);
            printFibonacci(count - 1);
        }
    }
    public static void main(String args[])
    {
        int count = 10;
        System.out.print(n1 + " " + n2); //printing 0 and 1
        printFibonacci(count - 2); //n-2 because 2 numbers are already printed
    }
}
```

https://compiler.javatpoint.com/opr/test.jsp?filename=FibonacciExample2

```
0 1 1 2 3 5 8 13 21 34
```

EXAMPLE: PRIME NUMBER

- Prime number in Java: Prime number is a number that is greater than 1 and divided by 1 or itself only. For example, 2, 3, 5, 7, 11, 13, 17.... are the prime numbers.

```java
public class PrimeExample
{
    public static void main(String args[])
    {
        int i, m = 0, flag = 0;
        int n = 3; //it is the number to be checked
        m = n / 2;
        if(n == 0 || n == 1)
        {
            System.out.println(n + " is not prime number");
        }
        else
        {
            for(i = 2; i <= m; i++)
            {
                if(n % i == 0)
                {
                    System.out.println(n + " is not prime number");
                    flag = 1;
                    break;
                }
            }
            if(flag == 0)
            {
                System.out.println(n + " is prime number");
            }
        }//end of else
    }
}
```

https://compiler.javatpoint.com/opr/test.jsp?filename=PrimeExample

```
3 is prime number
```

EXAMPLE: PALINDROME PROGRAM

- A palindrome number is a number that is same after reverse.
  - 545, 151, 34543, 343, 171, 48984 are the palindrome numbers.
- Algorithm
  - Get the number to check for palindrome
  - Hold the number in temporary variable
  - Reverse the number
  - Compare the temporary number with reversed number
  - If both numbers are same, print "palindrome number"
  - Else print "not palindrome number"

```java
class PalindromeExample
{
    public static void main(String args[])
    {
        int r, sum = 0, temp;
        int n = 454; //It is the number variable to be checked for palindrome

        temp = n;
        while(n > 0)
        {
            r = n % 10; //getting remainder
            sum = (sum * 10) + r;
            n = n / 10;
        }
        if(temp == sum)
            System.out.println("palindrome number ");
        else
            System.out.println("not palindrome");
    }
}
```

```
palindrome  number
```

```java
import java.util.*;
class PalindromeExample2
{
    public static void main(String args[])
    {
        String original, reverse = ""; // Objects of String class
        Scanner in = new Scanner(System.in);
        System.out.println("Enter a string/number to check if it is a palindrome");
        original = in.nextLine();
        int length = original.length();
        for ( int i = length - 1; i >= 0; i-- )
            reverse = reverse + original.charAt(i);
        if (original.equals(reverse))
            System.out.println("Entered string/number is a palindrome.");
        else
            System.out.println("Entered string/number isn't a palindrome.");
    }
}
```

EXAMPLE: CREATE AN OBJECT

- The object is a basic building block of an OOPs language. We cannot execute any Java program without creating an object.
- Create an object in Java
  - Using new Keyword: the most popular way. It allocates memory for the newly created object and returns the reference of that object to the memory.
  - Using clone() method
  - Using newInstance() method of the Class class
  - Using newInstance() method of the Constructor class
  - Using Deserialization

OBJECT VS CLASS

No.	Object	Class
1)	Object is an **instance** of a class.	Class is a **blueprint or template** from which objects are created.
2)	Object is a **real world entity** such as pen, laptop, mobile, bed, keyboard, mouse, chair etc.	Class is a **group of similar objects**.
3)	Object is a **physical** entity.	Class is a **logical** entity.
4)	Object is created through **new keyword** mainly e.g. Student s1=new Student();	Class is declared using **class keyword** e.g. class Student{}
5)	Object is created **many times** as per requirement.	Class is declared **once**.
6)	Object **allocates memory when it is created**.	Class **doesn't allocated memory when it is created**.
7)	There are **many ways to create object** in java such as new keyword, newInstance() method, clone() method, factory method and deserialization.	There is only **one way to define class** in java using class keyword.

```java
public class CreateObjectExample1
{
    void show()
    {
        System.out.println("Welcome to javaTpoint");
    }
    public static void main(String[] args)
    {
        //creating an object using new keyword
        CreateObjectExample1 obj = new CreateObjectExample1();
        //invoking method using the object
        obj.show();
    }
}
```

```
Welcome to javaTpoint
```

EXAMPLE: PRINT ASCII VALUE

- ASCII: American Standard Code for Information Interchange.
  - A 7-bit character set contains 128 (0 to 127) characters.
  - It represents the numerical value of a character.
  - Example: ASCII value of A is 65

- Two ways to print ASCII value:
  - Assigning a Variable to the int Variable
  - Using Type-Casting

```java
public class PrintAsciiValueExample1
{
    public static void main(String[] args)
    {
        // character whose ASCII value to be found
        char ch1 = 'a';
        char ch2 = 'b';
        // variable that stores the integer value of the character
        int asciivalue1 = ch1;
        int asciivalue2 = ch2;
        System.out.println("The ASCII value of " + ch1 + " is: " + asciivalue1);
        System.out.println("The ASCII value of " + ch2 + " is: " + asciivalue2);
    }
}
```

```
The ASCII value of a is: 97
The ASCII value of b is: 98
```

```java
public class PrintAsciiValueExample2
{
    public static void main(String[] String)
    {
        int ch1 = 'a';
        int ch2 = 'b';
        System.out.println("The ASCII value of a is: " + ch1);
        System.out.println("The ASCII value of b is: " + ch2);
    }
}
```

```
The ASCII value of a is: 97
The ASCII value of b is: 98
```

EXAMPLE: COUNT CHARACTERS

- Iterate through the string and count the characters.
  - STEP 1: START
  - STEP 2: DEFINE String *string = "The best of both worlds".*
  - STEP 3: SET *count =0.*
  - STEP 4: SET *i=0.* REPEAT STEP 5 to STEP 6 UNTIL *i<string.length*
  - STEP 5: IF *(string.charAt(i)!= ' ')* then *count =count +1.*
  - STEP 6: *i=i+1*
  - STEP 7: PRINT *count.*
  - STEP 8: END

```java
public class CountCharacter
{
    public static void main(String[] args)
    {
        String string = "The best of both worlds";
        int count = 0;

        //Counts each character except space
        for(int i = 0; i < string.length(); i++)
        {
            if(string.charAt(i) != ' ')
                count++;
        }

        //Displays the total number of characters present in the given string
        System.out.println("Total number of characters in a string: " + count);
    }
}
```

```
Total number of characters in a string: 19
```

EXAMPLE: COUNT PUNCTUATION CHARACTERS

- Algorithm
  - Define a string or read from the user.
  - Declare a variable to count the number of punctuations and initialized it with 0.
  - Match each character with the punctuation marks (!, . , ' , - , " , ? , ;). If any character in the string is matched, increase the count variable by 1.
  - Print the count variable that gives the total number of punctuations.

```java
public class CountPunctuation
{
    public static void main (String args[])
    {
        //Stores the count of punctuation marks
        int count = 0;
        String str = "He said, 'The mailman loves you.' I heard it with my own ears.";
        for (int i = 0; i < str.length(); i++)
        {
            //Checks whether given character is punctuation mark
            if(str.charAt(i) == '!' || str.charAt(i) == ',' || str.charAt(i) == ';' || str.
              charAt(i) == '.' ||  str.charAt(i) == '?' || str.charAt(i) == '-' ||
                    str.charAt(i) == '\'' || str.charAt(i) == '\"' || str.charAt(i) == ':')
            {
                count++;
            }
        }
        System.out.println("The number of punctuations exists in the string is: " + count);
    }
}
```

```
The number of punctuations exists in the string is: 5
```

EXAMPLE: REPLACE CHARACTERS

- Replace lower-case characters in a string to upper-case and vice versa.
  - STEP 1: START
  - STEP 2: DEFINE a string *str = "Great Power"*.
  - STEP 3: DEFINE *newstr* as StringBuffer object .
  - STEP 4: SET *i=0*. REPEAT STEP 5 to STEP 6 UNTIL *i<str.length()*.
  - STEP 5: IF lower-case character encountered then CONVERT in upper-case. ELSEIF upper-case character encountered then CONVERT in lower-case.
  - STEP 6: *i=i+1*
  - STEP 7: PRINT *newstr*.
  - STEP 8: END

```java
public class changeCase
{
    public static void main(String[] args)
    {

        String str1 = "Great Power";
        StringBuffer newStr = new StringBuffer(str1);

        for(int i = 0; i < str1.length(); i++)
        {

            //Checks for lower case character
            if(Character.isLowerCase(str1.charAt(i)))
            {
                //Convert it into upper case using toUpperCase() function
                newStr.setCharAt(i, Character.toUpperCase(str1.charAt(i)));
            }
            //Checks for upper case character
            else if(Character.isUpperCase(str1.charAt(i)))
            {
                //Convert it into upper case using toLowerCase() function
                newStr.setCharAt(i, Character.toLowerCase(str1.charAt(i)));
            }
        }
        System.out.println("String after case conversion : " + newStr);
    }
}
```

```
String after case conversion: gREAT pOWER
```

EXAMPLE: CHECK ROTATION OF A STRING

- Check whether string 2 is a rotation of string 1: Concatenate string 1 with string 1. If string 2 is present in concatenated string then, string 2 is rotation of string 1.
  - STEP 1: START
  - STEP 2: DEFINE String *str1 = "abcde", str2 = "deabc"*
  - STEP 3: IF length of *str1* not equals to *str2* then PRINT *"No"* else go to STEP 4
  - STEP 4: CONCATENATE *str1* with *str1*.
  - STEP 5: IF *str2* present in *str1* then PRINT "Yes" else PRINT "No".
  - STEP 6: END

```java
public class StringRotation
{
    public static void main(String[] args)
    {
        String str1 = "abcde", str2 = "deabc";

        if(str1.length() != str2.length())
        {
            System.out.println("Second string is not a rotation of first string");
        }
        else
        {
            //Concatenate str1 with str1 and store it in str1
            str1 = str1.concat(str1);
            //Check whether str2 is present in str1
            if(str1.indexOf(str2) != -1)
                System.out.println("Second string is a rotation of first string");
            else
                System.out.println("Second string is not a rotation of first string");
        }
    }
}
```

```
Second string is a rotation of first string
```

EXAMPLE: FILE HANDLING

- The File class is an abstract representation of file and directory pathname.
  - A pathname can be either absolute or relative.

- The File class have methods for working with directories and files:
  - creating new directories or files
  - deleting and renaming directories or files
  - listing the contents of a directory, etc.

- VERY USEFUL IN THIS CLASS

https://www.javatpoint.com/java-file-class

Modifier and Type	Method	Description
static File	createTempFile(String prefix, String suffix)	It creates an empty file in the default temporary-file directory, using the given prefix and suffix to generate its name.
boolean	createNewFile()	It atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist.
boolean	canWrite()	It tests whether the application can modify the file denoted by this abstract pathname.String[]
boolean	canExecute()	It tests whether the application can execute the file denoted by this abstract pathname.
boolean	canRead()	It tests whether the application can read the file denoted by this abstract pathname.
boolean	isAbsolute()	It tests whether this abstract pathname is absolute.
boolean	isDirectory()	It tests whether the file denoted by this abstract pathname is a directory.
boolean	isFile()	It tests whether the file denoted by this abstract pathname is a normal file.
String	getName()	It returns the name of the file or directory denoted by this abstract pathname.
String	getParent()	It returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory.
Path	toPath()	It returns a java.nio.file.Path object constructed from the this abstract path.
URI	toURI()	It constructs a file: URI that represents this abstract pathname.
File[]	listFiles()	It returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname
long	getFreeSpace()	It returns the number of unallocated bytes in the partition named by this abstract path name.
String[]	list(FilenameFilter filter)	It returns an array of strings naming the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
boolean	mkdir()	It creates the directory named by this abstract pathname.

```java
import java.io.*;
public class FileDemo
{
    public static void main(String[] args)
    {

        try
        {
            File file = new File("javaFile123.txt");
            if (file.createNewFile())
            {
                System.out.println("New File is created!");
            }
            else
            {
                System.out.println("File already exists.");
            }
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }

    }
}
```

```
New File is created!
```

```java
import java.io.*;
public class FileDemo2
{
    public static void main(String[] args)
    {

        String path = "";
        boolean bool = false;
        try
        {
            // createing  new files
            File file  = new File("testFile1.txt");
            file.createNewFile();
            System.out.println(file);
            // createing new canonical from file object
            File file2 = file.getCanonicalFile();
            // returns true if the file exists
            System.out.println(file2);
            bool = file2.exists();
            // returns absolute pathname
            path = file2.getAbsolutePath();
            System.out.println(bool);
            // if file exists
            if (bool)
            {
                // prints
                System.out.print(path + " Exists? " + bool);
            }
        }
        catch (Exception e)
        {
            // if any error occurs
            e.printStackTrace();
        }
    }
}
```

```
testFile1.txt
/home/Work/Project/File/testFile1.txt
true
/home/Work/Project/File/testFile1.txt Exists? true
```

```java
import java.io.*;
public class FileExample
{
    public static void main(String[] args)
    {
        File f = new File("/Users/sonoojaiswal/Documents");
        String filenames[] = f.list();
        for(String filename : filenames)
        {
            System.out.println(filename);
        }
    }
}
```

```
"info.properties"
"info.properties".rtf
.DS_Store
.localized
Alok news
apache-tomcat-9.0.0.M19
apache-tomcat-9.0.0.M19.tar
bestreturn_org.rtf
BIODATA.pages
BIODATA.pdf
BIODATA.png
struts2jars.zip
workspace
```

```java
import java.io.*;
public class FileExample
{
    public static void main(String[] args)
    {
        File dir = new File("/Users/Documents");
        File files[] = dir.listFiles();
        for(File file : files)
        {
            System.out.println(file.getName() + " Can Write: " + file.canWrite() + "Is Hidden: "
                + file.isHidden() + " Length: " + file.length() + " bytes");
        }
    }
}
```

```
"info.properties" Can Write: true Is Hidden: false Length: 15 bytes
"info.properties".rtf Can Write: true Is Hidden: false Length: 385 bytes
.DS_Store Can Write: true Is Hidden: true Length: 36868 bytes
.localized Can Write: true Is Hidden: true Length: 0 bytes
Alok news Can Write: true Is Hidden: false Length: 850 bytes
apache-tomcat-9.0.0.M19 Can Write: true Is Hidden: false Length: 476 bytes
apache-tomcat-9.0.0.M19.tar Can Write: true Is Hidden: false Length: 13711360 bytes
bestreturn_org.rtf Can Write: true Is Hidden: false Length: 389 bytes
BIODATA.pages Can Write: true Is Hidden: false Length: 707985 bytes
BIODATA.pdf Can Write: true Is Hidden: false Length: 69681 bytes
BIODATA.png Can Write: true Is Hidden: false Length: 282125 bytes
workspace Can Write: true Is Hidden: false Length: 1972 bytes
```

EXAMPLE: FILEINPUTSTREAM & FILEOUTPUTSTREAM

- Java FileInputStream class obtains input bytes from a file.
  - used for reading byte-oriented data (streams of raw bytes) such as image, audio etc.
  - can also read character-stream data.
  - For reading streams of characters, it is preferred to use FileReader.

- FileOutputStream write primitives values into a file.
  - Write byte-oriented and character-oriented data.
  - For character-oriented data, it is preferred to use FileWriter.

https://www.javatpoint.com/java-fileinputstream-class
https://www.javatpoint.com/java-fileoutputstream-class

Java FileInputStream class methods

Method	Description
int available()	It is used to return the estimated number of bytes that can be read from the input stream.
int read()	It is used to read the byte of data from the input stream.
int read(byte[] b)	It is used to read up to **b.length** bytes of data from the input stream.
int read(byte[] b, int off, int len)	It is used to read up to **len** bytes of data from the input stream.
long skip(long x)	It is used to skip over and discards x bytes of data from the input stream.
FileChannel getChannel()	It is used to return the unique FileChannel object associated with the file input stream.
FileDescriptor getFD()	It is used to return the FileDescriptor object.
protected void finalize()	It is used to ensure that the close method is call when there is no more reference to the file input stream.
void close()	It is used to closes the stream.

```java
import java.io.FileInputStream;
public class DataStreamExample
{
    public static void main(String args[])
    {
        try
        {
            FileInputStream fin = new FileInputStream("D:\\testout.txt");
            int i = fin.read();
            System.out.print((char)i);

            fin.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

Note: Before running the code, a text file named as "**testout.txt**" is required to be created. In this file, we are having following content:

```
Welcome to javatpoint.
```

After executing the above program, you will get a single character from the file which is 87 (in byte form). To see the text, you need to convert it into character.

Output:

```
W
```

```java
import java.io.FileInputStream;
public class DataStreamExample
{
    public static void main(String args[])
    {
        try
        {
            FileInputStream fin = new FileInputStream("D:\\testout.txt");
            int i = 0;
            while((i = fin.read()) != -1)
            {
                System.out.print((char)i);
            }
            fin.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

```
Welcome to javaTpoint
```

FileOutputStream class methods

Method	Description
protected void finalize()	It is used to clean up the connection with the file output stream.
void write(byte[] ary)	It is used to write **ary.length** bytes from the byte array to the file output stream.
void write(byte[] ary, int off, int len)	It is used to write **len** bytes from the byte array starting at offset **off** to the file output stream.
void write(int b)	It is used to write the specified byte to the file output stream.
FileChannel getChannel()	It is used to return the file channel object associated with the file output stream.
FileDescriptor getFD()	It is used to return the file descriptor associated with the stream.
void close()	It is used to closes the file output stream.

```java
import java.io.FileOutputStream;
public class FileOutputStreamExample
{
    public static void main(String args[])
    {
        try
        {
            FileOutputStream fout = new FileOutputStream("D:\\testout.txt");
            fout.write(65);
            fout.close();
            System.out.println("success...");
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

```
Success...
```

The content of a text file **testout.txt** is set with the data **A**.

testout.txt

```
A
```

```java
import java.io.FileOutputStream;
public class FileOutputStreamExample
{
    public static void main(String args[])
    {
        try
        {
            FileOutputStream fout = new FileOutputStream("D:\\testout.txt");
            String s = "Welcome to javaTpoint.";
            byte b[] = s.getBytes(); //converting string into byte array
            fout.write(b);
            fout.close();
            System.out.println("success...");
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

```
Success...
```

The content of a text file **testout.txt** is set with the data **Welcome to javaTpoint.**

testout.txt

```
Welcome to javaTpoint.
```

EXAMPLE: FILEREADER & FILEWRITER

- Both FileReader and FileWriter classes are character-oriented (for text file)
  - FileReader is used to read data from the file.
  - It returns data in byte format like FileInputStream class.

  - FileWriter is used to write character-oriented data to a file.
  - Unlike FileOutputStream class, no need to convert string into byte array.

https://www.javatpoint.com/java-filereader-class
https://www.javatpoint.com/java-filewriter-class

```java
import java.io.FileReader;
public class FileReaderExample
{
    public static void main(String args[])throws Exception
    {
        FileReader fr = new FileReader("D:\\testout.txt");
        int i;
        while((i = fr.read()) != -1)
            System.out.print((char)i);
        fr.close();
    }
}
```

Here, we are assuming that you have following data in "testout.txt" file:

```
Welcome to javaTpoint.
```

Output:

```
Welcome to javaTpoint.
```

```java
import java.io.FileWriter;
public class FileWriterExample
{
    public static void main(String args[])
    {
        try
        {
            FileWriter fw = new FileWriter("D:\\testout.txt");
            fw.write("Welcome to javaTpoint.");
            fw.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
        System.out.println("Success...");
    }
}
```

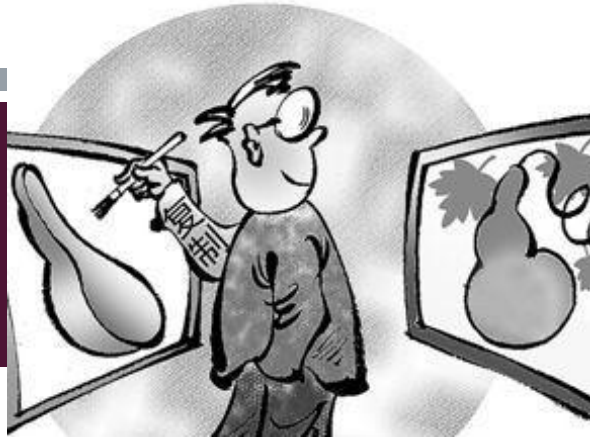
Output:

```
Success...
```

testout.txt:

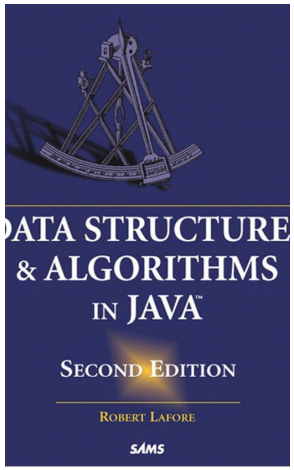
```
Welcome to javaTpoint.
```

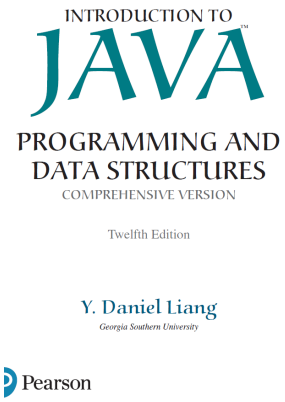
SUMMARY



- A quick introduction to Java
- Example + Pressure is the best way to learn a computer language quickly
- Project #1 is coming…

- Self study and practice are important
  - https://www.javatpoint.com/java-tutorial
  - Many good books, find one from our library





THANKS

PLEASE HELP INVITE CLASSMATES TO JOIN.