in28minutes

Learn Java Programming in 250 Steps

Learn the most popular programming language step by step with 200+ Hands on examples

Table of Contents

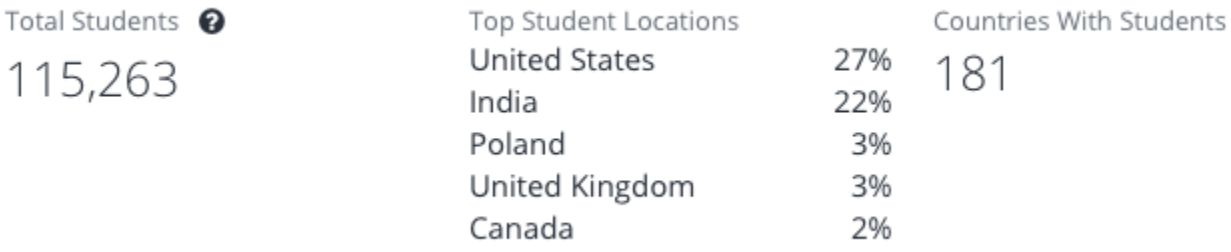
Congratulations

You have made a great choice in learning with in28Minutes. You are joining 100,000+ Learners learning everyday with us.

100,000+ Java beginners are learning from in28Minutes to become experts on APIs, Web Services and Microservices with Spring, Spring Boot and Spring Cloud.

About in28Minutes

How did in28Minutes get to 100,000 learners across the world?



We are focused on creating the awesome course (learning) experiences. Period.

An awesome learning experience? What's that?

You need to get insight into the in28Minutes world to answer that.

You need to understand "***The in28Minutes Way***"

- What are our beliefs?
- What do we love?
- Why do we do what we do?
- How do we design our courses?

Let's get started on "***The in28Minutes Way***"!

Getting Started

Recommended Versions

Tool/Framework/Language	Recommended Version	More Details
Java	Java 10+	We will use JShell - New feature in Java 9
Eclipse	Eclipse Java EE Oxygen	Eclipse 4.7.2 or Greater
Having Fun	Infinity!	Be prepared to work hard while having a lot of hands-on fun!

Installation

- Github : https://github.com/in28minutes...
- PDF : https://github.com/in28minutes...

Troubleshooting

- A 50 page troubleshooting guide with more than 200 Errors and Questions answered

Introduction To Java Programming With JShell

Steps

- Step 00 - Getting Started with Programming
- Step 01 - Introduction to Multiplication Table challenge
- Step 02 - Launch JShell
- Step 03 - Break Down Multiplication Table Challenge
- Step 04 - Java Expression - An Introduction
- Step 05 - Java Expression - Exercises
- Step 06 - Java Expression - Puzzles
- Step 07 - Printing output to console with Java
- Step 08 - Printing output to console with Java - Exercise Statements
- Step 09 - Printing output to console with Java - Exercise Solutions
- Step 10 - Printing output to console with Java - Puzzles
- Step 11 - Advanced Printing output to console with Java
- Step 12 - Advanced Printing output to console with Java - Exercises and Puzzles
- Step 13 - Introduction to Variables in Java
- Step 14 - Introduction to Variables in Java - Exercises and Puzzles
- Step 15 - 4 Important Things to Know about Variables in Java
- Step 16 - How are variables stored in memory?
- Step 17 - How to name a variable?
- Step 18 - Understanding Primitive Variable Types in Java
- Step 19 - Understanding Primitive Variable Types in Java - Choosing a Type
- Step 20 - Java Assignment Operator
- Step 21 - Java Assignment Operator - Puzzles on Increment, Decrement and Compound Assignment

- Step 23 - Java Conditionals and If Statement - Introduction

- Step 24 - Java Conditionals and If Statement - Exercise Statements
- Step 25 - Java Conditionals and If Statement - Exercise Solutions
- Step 26 - Java Conditionals and If Statement - Puzzles
- Step 27 - Java For Loop to Print Multiplication Table - Introduction
- Step 28 - Java For Loop to Print Multiplication Table - Exercise Statements
- Step 29 - Java For Loop to Print Multiplication Table - Exercise Solutions
- Step 30 - Java For Loop to Print Multiplication Table - Puzzles
- Step 31 - Programming Tips : JShell - Shortcuts, Multiple Lines and Variables TODO Move up
- Step 32 - Getting Started with Programming - Revise all Terminology

Exercises

Expressions

- Write a Java expression to calculate the number of minutes in a day.
- Write a Java expression to calculate the number of seconds in a day.

Printing Output to Console

- Print `Hello World` onto the console.
- Print `5 * 3` , as is.
- Print the calculated value of `5 * 3` .
- How do we now approach the Multiplication Table print problem, for the number `5` ?
- Print the number of seconds in a day, using the `System.out.println` method.
- Do a syntax revision for the code that you write for each of the above exercises. In your code, identify the following elements:
  - Numeric and string literals
  - Expressions
  - Operators
  - Operands
  - Method calls

Advanced Printing Output to Console

- Print the following output by passing 4 values 5, ,6, 7 and sum of the numbers: `5`

- `  + 6 + 7 = 18`

Variables

- Create three integer variables `a`, `b` and `c`.
  - Write a statement to print the sum of these three variables.
  - Change the value of `a`, and then print this sum.
  - Then again, change the value of `b` and print this sum.

If Statement

- Create four integer variables `a`, `b`, `c` and `d`. - Write an `if` statement to print if the sum of `a` and `b` is greater than the sum of `c` and `d`.
- Store three numerical values as proposed angles of a triangle in integer variables `angle1`, `angle2` and `angle3`. Create an `if` statement to state whether these three angles together can form a triangle. *Hint: A triangle is a closed geometric figure with three angles, whose sum must exactly equal* `180 degrees`.
- Have a variable store an integer. Create an `if` statement to find out if it's an even number.*Hint: Use operator ###code#<.*

For Loop

- Repeat the entire process at arriving at the Multiplication Table Print problem, now for the number `7`. Start with fresh JShell session, if you're still using an existing one for quite some time (Rinse, and repeat!).
- Use the final solution to print Multiplication Tables for `6` and `10`.
- Print the integers from `1` to `10`.
- Print the integers from `10` to `1`.
- Print the squares of the integers from `1` to `10`.
- Print the squares of the first `10` even integers.
- Print the squares of the first `10` odd integers.

Code Snippets

```
3*4
System.out.println(3*4)
```

```
System.out.println("5 * 2 = 10")
3.0/2
System.out.println("5 * 2")
System.out.println(5 * 2)
System.out.println("Hello World")
System.out.println(5*3)
System.out.println("5 * 3")
System.out.println(5*3) System.out.println("5 * 1 = 5")
System.out.println("5 * 2 = 5")
System.out.println("5 * 3 = 5")
System.out.println("24 * 60 * 60")
System.out.println(24 * 60 * 60)
System.out.println("Hello World")
System.out.println("Hello     World")
System.out.println(24   *    60    *    60)
System.out.println("Hello World")
System.out.println("HelloWorld")
System.out.println("hello world")
System.out.println("hello \"world")
System.out.println("hello \"world")
System.out.println("hello n world")
System.out.println("hello \n world")
System.out.println("hello \nworld")
System.out.println("hello\nworld")
System.out.println("hello\tworld")
System.out.println("hello \\ world")
System.out.println("hello \\\\ world")
Math.random()
System.out.println("hello \n world")
Math.min(23,45)
Math.min(23,4)
Math.max(23,4)
System.out.println("5 * 2 = 5")
System.out.println("5 * 2 = 10")
5*2
```

```
System.out.printf("5 * 2 = 10")
System.out.printf("5 * 2 = 10").println()
System.out.printf("5 * 2 = 10 %d", 5*2).println()
System.out.printf("5 * 2 = %d", 5*2).println()
System.out.printf("%d %d %d", 5, 7, 5 * 7).println()
System.out.printf("%d * %d = %d", 5, 7, 5 * 7).println()
System.out.printf("%d + %d + %d = %d", 5, 6, 7, 5 + 6 +
7).println()
System.out.printf("%d + %d + %d = %d", 5, 6, 7).println()
System.out.printf("%d + %d + %d", 5, 6, 7).println()
System.out.printf("%d + %d + %d", 5, 6).println()
System.out.printf("%d + %d + %d", 5, 6, 7, 8).println()
System.out.printf("Print %s", "Testing").println()
System.out.printf("%d + %d + %d", 5.5, 6.5, 7.5).println()
System.out.printf("%f + %f + %f", 5.5, 6.5, 7.5).println()
System.out.printf("5 * 2 = 10")
System.out.printf("%d * %d = %d", 5, 2, 5 * 2).println()
System.out.println("5 * 2 = 10")
System.out.printf("%d * %d = %d", 5, 2, 5 * 2).println()
System.out.printf("%d * %d = %d", 5, 1, 5 * 1).println()
System.out.printf("%d * %d = %d", 5, 2, 5 * 2).println()
System.out.printf("%d * %d = %d", 5, 3, 5 * 3).println()
System.out.printf("%d * %d = %d", 5, 4, 5 * 4).println()
number = 11
number
number = 12
number
number2 = 100
System.out.printf("%d * %d = %d", 5, 4, 5 * 4).println()
System.out.printf("%d * %d = %d", 5, i, 5 * i).println()
i
5 * i
i = 2
System.out.printf("%d * %d = %d", 5, i, 5 * i).println()
i = 3
System.out.printf("%d * %d = %d", 5, i, 5 * i).println()
```

```
i = 10
System.out.printf("%d * %d = %d", 5, i, 5 * i).println()
System.out.printf("a + b + c = a+b+c").println()
System.out.printf("%d + %d + %d = %d", a, b, c
,a+b+c).println()
a = 50 System.out.printf("%d + %d + %d = %d", a, b, c
,a+b+c).println()
b = 60
System.out.printf("%d + %d + %d = %d", a, b, c
,a+b+c).println()
int newVariable;
newVariable
int undeclaredVariable;
undeclaredVariable
5 * undeclaredVariable
a = 100
a = c
int noOfGoals;
int NoOfGoals;
int score;
short s;
float f = 4.0f;
float f2 = 4.5f;
double dbl = 4.5;
i = j
i
j
i = j
i = j * 2
i = i * 2
i = i + i
i = i - i
i = i - 1
i++
i
```

```
i++
i
i--
number = number + 1
number++ number--
++number
--number
number--
number
i = i + 2
i += 2
i -= 1
i *= 5
i
i /= 4
i %= 2
;
long l = 6_000_000_000l;
short numberOfGoals;
numberOfGoals++
numberOfGoals
numberOfGoals++
numberOfGoals
long populationOfTheWorld;
double average;
char ch = 'A';
ch = 'B'
char grade = 'C';
grade = 'A'
grade
boolean isEven;
isEven = true
boolean isPrime;
boolean isItRainingToday;
boolean areYouEnjoyingTheCourse;
```

```
areYouEnjoyingTheCourse = true
System.out.printf("%d * %d = %d", 5, i, 5*i);
System.out.printf("%d * %d = %d", 5, i, 5*i).println()
System.out.printf("%d * %d = %d", 5, i, 5*i).println()
System.out.printf("%d * %d = %d", 5, i, 5*i).println()
i = 7
System.out.printf("%d * %d = %d", 5, i, 5*i).println()
i = 10
i < 5
i > 5
i <= 5
i <= 10
i >= 10
System.out.println("i is less than 5");
i
if (i<5)
   System.out.println("i is less than 5");
i
i = 4
if (i<5)
   System.out.println("i is less than 5");
int number1 = 5;
int number2 = 7;
if (number2>number1)
System.out.println("number2 is greater than number1");
number2 = 3
if (number2>number1)
System.out.println("number2 is greater than number1");
int a = 1;
int b = 2;
int c = 3;
int d = 1;
if(a+b > c +d)
System.out.println("a+b is greater than c+d");
a = 6
if(a+b > c +d)
```

```
System.out.println("a+b is greater than c+d");
if ( a + b > c + d )
    System.out.println("a+b is greater than c+d"); int
angle1 = 20;
int angle2 = 60;
int angle3 = 50;
if(angle1 + angle2 + angle3 == 180)
    System.out.println("Valid Triangle");
angle3 += 50
angle3
angle1
angle2
if(angle1 + angle2 + angle3 == 180)
    System.out.println("Valid Triangle");
int number = 10;
number % 2
9 % 2
8 % 2
if (number % 2 == 0)
System.out.println("number is even");
if (number % 2 == 0)
System.out.println("number is even");
number = 9
if (number % 2 == 0)
System.out.println("number is even");
i > 5
i = 5
i == 5
i == 6
if(i==5)
System.out.println("i is odd");
if(i==5)
System.out.println("i is odd");
 System.out.println("i is prime");
i == 6
i = 6
```

```
if(i==5)
System.out.println("i is odd");
  System.out.println("i is prime");
if(i==5) {
System.out.println("i is odd");
System.out.println("i is prime");
}
if(i==5) {
System.out.println("i is prime");
}
System.out.printf("%d * %d = %d", 5, i, 5*i).println()
System.out.printf("%d * %d = %d", 5, i, 5*i).println()
i = i + 1
System.out.printf("%d * %d = %d", 5, i, 5*i).println()
i = i + 1
System.out.printf("%d * %d = %d", 5, i, 5*i).println()
for( i =1; i<=10; i++) {
   System.out.printf("%d * %d = %d", 5, i, 5*i).println();
}
for( i =1; i<=10; i++) {
   System.out.printf("%d * %d = %d", 6, i, 6*i).println();
}
for( i =1; i<=10; i++) {
   System.out.printf("%d * %d = %d", 7, i, 7*i).println();
}
int table = 7;
for( i =1; i<=10; i++) {
   System.out.printf("%d * %d = %d", table, i,
table*i).println();
}
table = 8
for( i =1; i<=10; i++) {
   System.out.printf("%d * %d = %d", table, i,
table*i).println();
}
table = 8
```

```
for( i =1; i<=10; i++)  {
   System.out.printf("%d * %d = %d", table, i,
table*i).println();
}
table = 9
for( i =1; i<=10; i++) {
   System.out.printf("%d * %d = %d", table, i,
table*i).println();
}
for(i =1; i<=10; i++) {
   System.out.printf("%d", i).println();
}
for(i=10; i<=1  ;i--) {
   System.out.printf("%d", i).println();
}
for(i=10; i>=1  ;i--) {
   System.out.printf("%d", i).println();
}
for(i=10; i>=1  ;i = i - 2) {
   System.out.printf("%d", i).println();
}
for(i=9; i>=1  ;i = i - 2) {
   System.out.printf("%d", i).println();
}
for(i =1; i<=10; i++) {
   System.out.printf("%d", i * i).println();
}
for(i =2; i<=10; i = i + 2) {
   System.out.printf("%d", i * i).println();
}
for(i =2; i<=20; i = i + 2) {
   System.out.printf("%d", i * i).println();
}
for(i =1; i<=10; i++) {
   System.out.printf("%d", (2 * i) * (2 * i) ).println();
}
```

```
for(i =1; i<=20; i = i + 2) {
   System.out.printf("%d", i * i).println();
}
int i = 1;
for ( ;i<=10 ;i++ );
i
for (i=1 ;i<=10 ;i++ );
int j;
for (i=1, j=2 ;i<=10 ;i++, j++ );
i
j
for (i=1, j=2 ;i<=10 ;i++, j--);
i
j
for(;;);
for(int i=1; i<=10; i++) {
System.out.println("No 1");
System.out.println("No 2");
}
for(int i=1; i<=10; i++) {
System.out.println("No 1");
System.out.println("No 2");
}
   System.out.println("i is less than 5");
```

Introduction To Methods - Multiplication Table

Steps

- Step 00 - Section 02 - Methods - An Introduction
- Step 01 - Your First Java Method - Hello World Twice and Exercise Statements
- Step 02 - Introduction to Java Methods - Exercises and Puzzles
- Step 03 - Programming Tip - Editing Methods with JShell
- Step 04 - Introduction to Java Methods - Arguments and Parameters
- Step 05 - Introduction to Java Method Arguments - Exercises
- Step 06 - Introduction to Java Method Arguments - Puzzles and Tips
- Step 07 - Getting back to Multiplication Table - Creating a method
- Step 08 - Print Multiplication Table with a Parameter and Method Overloading
- Step 09 - Passing Multiple Parameters to a Java Method
- Step 10 - Returning from a Java Method - An Introduction
- Step 11 - Returning from a Java Method - Exercises
- Step 99 - Methods - Section Review

Exercises

Methods - Basics

- Write and execute a method named `sayHelloWorldThrice` to print `Hello World` thrice.
- Write and execute a method that prints the following four statements:
  - `I've created my first variable`
  - `I've created my first loop`
-
  -

- `I've created my first method`
  - `I'm excited to learn Java`

Method Parameters

1. Write a method `printNumbers(int n)` that prints all successive integers from `1` to `n`.
2. Write a method `printSquaresOfNumbers(int n)` that prints the squares of all successive integers from `1` to `n`.

Return value from Methods

1. Write a method that returns the sum of three integers.
2. Write a method that takes as input two integers representing two angles of a triangle, and computes the third angle. *Hint: The sum of the three angles of a triangle is 180 degrees.*

Code Snippets

```
3 * 4
$2
$2 * 3
for( i =1; i<=10; i++) {
System.out.printf("%d * %d = %d", table, i,
table*i).println();
}
for( i =1; i<=10; i++) {
    System.out.printf("%d * %d = %d", table, i,
table*i).println();
}
for( i =1; i<=10; i++) {
    System.out.printf("%d * %d = %d", table, i,
table*i).println();
}
for( i =1; i<=10; i++) {
   System.out.printf("%d * %d = %d", table, i,

 table*i).println();
```

```
}
int table = 7;
for( i =1; i<=10; i++) {
System.out.printf("%d * %d = %d", table, i,
table*i).println(); }
table = 8
for( i =1; i<=10; i++) {

System.out.printf("%d * %d = %d", table, i,
table*i).println();
}
System.out.println("Hello World")
System.out.println("Hello World")
System.out.println("Hello World")
System.out.println("Hello World");
 System.out.println("Hello World");
System.out.println("Hello World");
 System.out.println("Hello World");
sayHelloWorldTwice()
sayHelloWorldTwice()
void sayHelloWorldThrice(){
 System.out.println("Hello World");
 System.out.println("Hello World");
 System.out.println("Hello World");
}
sayHelloWorldThrice()
printLearningExperience()
printLearningExperience()
void NameOfMethod(){
}
void nameOfMethod(){
}
void sayHelloWorldTwice(){
   System.out.println("HelloWorld");
   System.out.println("HelloWorld");
```

```
} sayHelloWorldTwice()

void printLearningExperience() {
System.out.println("I've created sdf first variable");
System.out.println("I've created fsadjf first method");
System.out.println("I've created fsajdfl first loop");
System.out.println("I'm excited fasdflkjfskfsd learn
Java");
}
printLearningExperience()

sayHelloWorldThrice() sayHelloWorldTwice() sayHelloWorld(1)
sayHelloWorld(1)
sayHelloWorld(2)
sayHelloWorld(2)
sayHelloWorld(4)
sayHelloWorld(6)
void sayHelloWorld(int noOfTimes) {
        for(int i=1; i<=noOfTimes; i++) {
                System.out.println("Hello World");
        }
}
sayHelloWorld(6)
sayHelloWorld(3)
sayHelloWorld(2)
void printNumbers(int n) {
    for(int i=1; i<=n; i++) {
            System.out.println(i);
    }
}
printNumbers(10)
printNumbers(3)
void printSquaresOfNumbers(int n) {
    for(int i=1; i<=n; i++) {
         System.out.println(i*i);
    }
} printSquaresOfNumbers(5)
printSquaresOfNumbers(3)
```

```
sayHelloWorld(4)

for( i =1; i<=10; i++) {
        System.out.printf("%d * %d = %d", table, i,
table*i).println();
}
int i; for(i=1;i<=10;i++)
    System.out.printf("%d * %d = %d", 5, i, 5 *
i).println(); for(i=1;i<=10;i++)

 {
    System.out.printf("%d * %d = %d", 5, i, 5 *
i).println();
}
void printMultiplicationTable() {
        for(int i=1;i<=10;i++) {
                System.out.printf("%d * %d = %d", 5, i, 5 *
i).println();
        }
}
printMultiplicationTable()

void printMultiplicationTable(int table) {
        for(int i=1;i<=10;i++) {
                System.out.printf("%d * %d = %d", table, i,
table * i).println();
        }
}

printMultiplicationTable(6)
printMultiplicationTable(7)
printMultiplicationTable()
printMultiplicationTable(6)
Math.max(1,2)
void sum(int firstNumber,int secondNumber)  {
     int sum = firstNumber +
```

```
 secondNumber;
     System.out.println(sum);

}
sum(5, 10) void sum(int firstNumber,int secondNumber, int

 thirdNumber) {
     int sum = firstNumber + secondNumber + thirdNumber;
     System.out.println(sum); }
sum(5, 10, 15) Math.max(4,5)
$110 int max = Math.max(15, 25);

max
sum(1, 10)
int sumOfTwoNumbers(int firstNumber, int secondNumber) {
    int sum = firstNumber + secondNumber;
    return sum;
}
sumOfTwoNumbers(1,1)
int sum = sumOfThreeNumbers(2,3,4);
calculateThirdAngle(20, 50)
int sumOfThreeNumbers(int firstNumber, int secondNumber,
int thirdNumber) {
      int sum = firstNumber + secondNumber + thirdNumber;
      return sum;
}
sumOfThreeNumbers(1,2,3)
sumOfThreeNumbers(15,2,3)
int calculateThirdAngle(int angle1, int angle2) {
       int angle3 = 180 - (angle1 + angle2);
       return angle3;
}
calculateThirdAngle(20, 20)
calculateThirdAngle(20, 20)
```

Introduction To Java Platform

Steps

- Step 00 - Section 03 - Overview Of Java Platform - Section Overview
- Step 01 - Overview Of Java Platform - An Introduction - java, javac, bytecode and JVM
- Step 02 - Java Class and Object - First Look
- Step 03 - Create a method in a Java class
- Step 04 - Create and Compile Planet.java class
- Step 05 - Run Planet calss with Java - Using a main method
- Step 06 - Play and Learn with Planet Class
- Step 07 - JDK vs JRE vs JVM

Code Snippets (Including Output)

```
jshell> System.out.println("I love JShell");
I love JShell

jshell> class Country {
   ...> }
|  created class Country

jshell> Country india = new Country()
india ==> Country@6e06451e

jshell> Country usa = new Country()
usa ==> Country@6e1567f1

jshell> class Planet
```

```
 {
   ...> }
|  created class Planet

jshell> Planet planet = new Planet()
planet ==> Planet@56ef9176

jshell> Planet earth = new Planet()

earth ==> Planet@1ed4004b

jshell> Planet venus = new Planet()
venus ==> Planet@25bbe1b6

jshell> void printMultiplicationTable() {
   ...>     for(int i=1;i<=10;i++) {
   ...>         System.out.printf("%d * %d = %d", 5, i, 5 *
i).println();
   ...>     }
   ...> }
|  created method printMultiplicationTable()

jshell> void printMultiplicationTable(int table) {
   ...>     for(int i=1;i<=10;i++) {
   ...>         System.out.printf("%d * %d = %d", table, i,
table * i).println();
   ...>     }
   ...> }
|  created method printMultiplicationTable(int)

jshell>

jshell> /methods
|    void |    void printMultiplicationTable()
|    void printMultiplicationTable(int)
```

```
jshell>

jshell> class Planet {
   ...> }
|  modified class Planet

jshell> Planet earth = new Planet()
earth ==> Planet@73846619

jshell> Planet venus = new Planet()
venus ==> Planet@4bec1f0c

jshell> class Planet {
   ...>      void revolve() {
   ...>           System.out.println("Revolve");
   ...>      }
   ...> }
|  replaced class Planet
|    update replaced variable planet, reset to null
|    update replaced variable earth, reset to null
|    update replaced variable venus, reset to null

jshell> Planet earth = new Planet()
earth ==> Planet@192b07fd

jshell> Planet venus = new Planet()
venus ==> Planet@64bfbc86

jshell> Planet.revolve()
|  Error:
|  non-static method revolve() cannot be referenced from a
static context
|  Planet.revolve()
|  ^------------^

jshell> earth.revolve()
Revolve
```

```
jshell> venus.revolve()
Revolve

jshell> class Country {
   ...>    void comingSoon() {
   ...>       System.out.println("Coming Soon");
   ...>

 }
   ...> }
|  replaced class Country
|    update replaced variable country, reset to null
|    update replaced variable india, reset to null
|    update replaced variable usa, reset to null

jshell> Country india = new Country()
india ==> Country@60c6f5b

jshell> Country netherlands = new Country()
netherlands ==> Country@3c0f93f1

jshell> india.comingSoon()
Coming Soon

jshell> netherlands.comingSoon()
Coming Soon

jshell> /list Country

  27 : class Country {
          void comingSoon() {
             System.out.println("Coming Soon");
          }
       }

jshell> /list
```

```
 Planet

  22 : class Planet

 {
            void revolve() {
                 System.out.println("Revolve");
            }
       }

jshell>
```

/03-IntroductionToJavaPlatform/Planet.java

```
class Planet {

    void revolve() {
         System.out.println("Revolve");
    }

    public static void main(String[] args) {
        Planet earth = new Planet();
        earth.revolve();
    }
}
```

Introduction To Eclipse-First Java Project

Steps

- Step 00 - Intro to Section and Installing Eclipse
- Step 01 - Creating a New Java Project with Eclipse
- Step 02 - Your first Java class with Eclipse
- Step 03 - Writing Multiplication Table Java Program with Eclipse
- Step 04 - Adding more methods for Multiplication Table Program
- Step 05 - Programming Tip 1 : Refactoring with Eclipse
- Step 06 - Programming Tip 2 : Debugging with Eclipse
- Step 07 - Programming Tip 3 : Eclipse vs JShell - How to choose?

Code Examples

/04-IntroductionToEclipse-FirstJavaProject/src/com/in28minutes/firstjavaproject/HelloWorld.java

```
package com.in28minutes.firstjavaproject;

public class HelloWorld {

	public static void main(String[] args) {

	}

}
```

/04-IntroductionToEclipse-FirstJavaProject/src/com/in28minutes/firstjavaproject/KeyboardShortcuts.java

```
package com.in28minutes.firstjavaproject;

public class KeyboardShortcuts {
	public static void main(String[] args) {
		int i = 0;
		System.out.println(i);
	}
}
```

/04-IntroductionToEclipse-FirstJavaProject/src/com/in28minutes/firstjavaproject/MultiplicationTable.java

```
package com.in28minutes.firstjavaproject;

public class MultiplicationTable {

	void print() {
		print(5);
	}

	void print(int table) {
		print(table, 1, 10);
	}

	void print(int table, int from, int to) {
		for (int i = from; i <= to; i++) {
			System.out.printf("%d X %d = %d",
table, i, table * i).println();
		}
```

```
        }

}
```

/04-IntroductionToEclipse-FirstJavaProject/src/com/in28minutes/firstjavaproject/MultiplicationTableRunner.java

```
package com.in28minutes.firstjavaproject;
 public class MultiplicationTableRunner {

        public static void main(String[] args) {
                MultiplicationTable table = new
MultiplicationTable();
                table.print();

                //table.print(6);
                //table.print(6, 11, 20);
        }

}
```

/04-IntroductionToEclipse-FirstJavaProject/src/com/in28minutes/firstjavaproject/MutliplicationTableBeforeRefactoring.java

```
package com.in28minutes.firstjavaproject;

public class MutliplicationTableBeforeRefactoring {
        void print() {
                for (int i = 1; i <= 10; i++) {
                        System.out.printf("%d X %d = %d",
5, i, 5 * i).println();
                }
        }
```

```
        void print(int table)

 {
                for (int i = 1; i <= 10; i++) {
                        System.out.printf("%d X %d = %d",
table, i, table * i).println();
                }
        }

        void print(int table, int from, int to) {
                for (int i = from; i <= to; i++) {
                        System.out.printf("%d X %d = %d",
table, i, table * i).println();
                }
        }
}
```

Introduction To Object Oriented Programming

Steps

- Step 00 - Introduction to Object Oriented Programming - Section Overview
- Step 01 - Introduction to Object Oriented Programming - Basics
- Step 02 - Introduction to Object Oriented Programming - Terminology - Class, Object, State and Behavior
- Step 03 - Introduction to Object Oriented Programming - Exercise - Online Shopping System and Person
- Step 04 - Create Motor Bike Java Class and a couple of objects
- Step 05 - Exercise Solutions - Book class and Three instances
- Step 06 - Introducing State of an object with speed variable
- Step 07 - Understanding basics of Encapsulation with Setter methods
- Step 08 - Exercises and Tips - Getters and Generating Getters and Setters with Eclipse
- Step 09 - Puzzles on this and initialization of member variables
- Step 10 - First Advantage of Encapsulation
- Step 11 - Introduction to Encapsulation - Level 2
- Step 12 - Encapsulation Exercises - Better Validation and Book class
- Step 13 - Introdcution to Abstraction
- Step 14 - Introduction to Java Constructors
- Step 15 - Introduction to Java Constructors - Exercises and Puzzles
- Step 16 - Introduction to Object Oriented Programming - Conclusion

Exercises

- In each of the following systems, identify the basic entities involved, and organize

- them using object oriented terminology:
  - Online Shopping System
  - Person

- Provide Better Encapsulation and Validation for Book and MotorBike class
- Create a constructor for Book class and create three instances

Code Examples

/05-IntroductionToObjectOrientedProgramming/src/com/in28minutes/oops/Book.java

```
package com.in28minutes.oops;

public class Book {

	private int noOfCopies;

	public Book(int noOfCopies) {
		this.noOfCopies = noOfCopies;
	}

	public void setNoOfCopies(int noOfCopies) {
		if (noOfCopies > 0)
			this.noOfCopies = noOfCopies;
	}

	public void increaseNoOfCopies(int howMuch) {
		setNoOfCopies(this.noOfCopies + howMuch);
	}

	public void decreaseNoOfCopies(int howMuch) {
		setNoOfCopies(this.noOfCopies - howMuch);
	}

}
```

/05-IntroductionToObjectOrientedProgramming/src/com/in28minutes/oops/BookRunner.java

```
package com.in28minutes.oops;
 public class BookRunner {

        public static void main(String[] args) {
                // Create a new class called Book
                // Create three instances
                Book artOfComputerProgramming = new
Book(100);
                Book effectiveJava = new Book(50);
                Book cleanCode = new Book(40);


artOfComputerProgramming.setNoOfCopies(100);
                effectiveJava.setNoOfCopies(50);
                cleanCode.setNoOfCopies(45);
        }

}
```

/05-IntroductionToObjectOrientedProgramming/src/com/in28minutes/oops/MotorBike.java

```
package com.in28minutes.oops;

public class MotorBike {
        //state
        private int speed; //member variable
```

```
        //constructors
        MotorBike() {
                this(5);
        }

        MotorBike(int speed)  {
                this.speed = speed;
        }

        //behavior
        public int getSpeed() {
                return speed;
        }

        public void setSpeed(int speed) {
                if(speed > 0 )
                        this.speed = speed;
        }

        public void increaseSpeed(int howMuch) {
                setSpeed(this.speed + howMuch);
        }

        public void decreaseSpeed(int howMuch) {
                setSpeed(this.speed - howMuch);
        }

        void start() {
                System.out.println("Bike Started");
        }
}
```

/05-IntroductionToObjectOrientedProgramming/src/com/in28minutes/oops/MotorBikeRunner.java

```
package com.in28minutes.oops;

public class MotorBikeRunner {

	public static void main(String[] args)  {

		MotorBike ducati = new MotorBike(100);

		MotorBike honda = new MotorBike(200);

		MotorBike somethingElse = new MotorBike();

		System.out.println(ducati.getSpeed());

		System.out.println(honda.getSpeed());


System.out.println(somethingElse.getSpeed());


		ducati.start();

		honda.start();

		//ducati.setSpeed(100);

		ducati.increaseSpeed(100);

		honda.increaseSpeed(100);

		ducati.decreaseSpeed(250);

		honda.decreaseSpeed(250);
```

```
        System.out.println(ducati.getSpeed());

                System.out.println(honda.getSpeed());
        }
 }

// Create a new class called Book
// Create three instances
// Art Of Computer Programming
// Effective Java
// Clean Code
```

/05-IntroductionToObjectOrientedProgramming/entireoutput-constructor-puzzles.txt

```
Last login: Mon Jan 29 10:33:44 on ttys000
Rangas-MacBook-Pro:~ rangaraokaranam$ jshell
|  Welcome to JShell -- Version 9.0.1
|  For an introduction type: /help intro

jshell> class Cart {
   ...> };
|  created class Cart

jshell> Cart cart1 = new Cart();
cart1 ==> Cart@3f49dace

jshell> class Cart {
   ...>     Cart() {
   ...>     }
   ...> };
|  replaced class Cart
|    update replaced variable cart1, reset to null

jshell> Cart cart1 = new Cart();
```

```
cart1 ==> Cart@59494225

jshell> class Cart {
   ...>     Cart()  {
   ...>         System.out.println("Constructor is
called");
   ...>     }
   ...> }
|  modified class Cart

jshell> Cart cart1 = new Cart();
Constructor is called
cart1 ==> Cart@6e1567f1

jshell>
```

Primitive Data Types And Alternatives

Steps

- Step 00 - Primitive Data Types in Depth - Section Overview
- Step 01 - Basics about Java Integer Data Types - Casting, Operators and More
- Step 02 - Java Integer Data Types - Puzzles - Octal, Hexadecimal, Post and Pre increment
- Step 03 - Java Integer Data Types - Exercises - BiNumber - add, multiply and double
- Step 04 - Java Floating Point Data Types - Casting , Conversion and Accuracy
- Step 05 - Introduction to BigDecimal Java Class
- Step 06 - BigDecimal Puzzles - Adding Integers
- Step 07 - BigDecimal Exercises - Simple Interest Calculation
- Step 08 - Java Boolean Data Type - Relational and Logical Operators
- Step 09 - Java Boolean Data Type - Puzzles - Short Circuit Operators
- Step 10 - Java Character Data Type char - Representation and Conversion
- Step 11 - Java char Data Type - Exercises 1 - isVowel
- Step 12 - Java char Data Type - Exercises 2 - isDigit
- Step 13 - Java char Data Type - Exercises 3 - isConsonant, List Upper Case and Lower Case Characters
- Step 14 - Primitive Data Types in Depth - Conclusion

Exercises

Big Decimal

- Calculate formula for Simple Interest Formula
  - Total Amount = principal + principal * interest * noOfYears;

```
SimpleInterestCalculator calculator
```

```
 =
        new SimpleInterestCalculator("4500.00", "7.5");
 BigDecimal totalValue  =
        calculator.calculateTotalValue(5);// 5 years
 System.out.println(totalSum);
```

char data type

Implement MyChar class

```
MyChar myChar = new MyChar('c');
System.out.println(myChar.isVowel());
        //'a', 'e', 'i', 'o', 'u' and Capitals
System.out.println(myChar.isDigit());
System.out.println(myChar.isAlphabet());
MyChar.printLowerCaseAlphabets();
MyChar.printUpperCaseAlphabets();
```

Code Examples

/06-PrimitiveDataTypesAndAlternatives/commands.txt

```
Byte.SIZE
Byte.BYTES
Byte.MAX_VALUE
Byte.MIN_VALUE
Short.BYTES
Integer.BYTES
Long.BYTES
Integer.MAX_VALUE
Short.MAX_VALUE
Byte.SIZE
Byte.MIN_VALUE
Byte.MAX_VALUE
Short.BYTES


Integer.BYTES
```

```
Long.BYTES
Integer.MAX_VALUE
byte c = 13;
long l = 500000000001; i = (int) l
l = i
int eight = 010;
int sixteen = 0x10;
int fifteen = 0XF; int big = 0XBBAACC; Short.MAX_VALUE

short s = (short) i;
int i1 = s;
i
i
i
i
34.5
34.56789
double dbl = 34.5678;
float f2 = (float)dbl;
dbl++
dbl--
dbl % 5
float f = i;
34.56789876 + 34.2234
number1.add(number2);
number1
BigDecimal number3 = number1.add(number2);
number1
BigDecimal number1 = new BigDecimal("34.56789876");
BigDecimal number10 = new BigDecimal(34.2234);
BigDecimal number11 = new BigDecimal("34.56789876");
number10.add(number11)
number10.multiply(number11)

BigDecimal number = new BigDecimal("11.5");

BigDecimal number2 = new BigDecimal("23.45678");
```

```
number.add(number2)
number.add(new BigDecimal(i))
number.multiply(new BigDecimal(i)) number.divide(new
BigDecimal(100))
number.divide(new BigDecimal("100.01234"))
number.divide(new BigDecimal("100.012"))
number.divide(new BigDecimal("100.12")) number.divide(new
BigDecimal("100.1")) number.divide(new BigDecimal("1001"))

number.divide(new BigDecimal("100"))
boolean isValue = false;
i > 7
i >= 7
i < 7
i <= 7
i == 6
i == 7
i == 8
i = 8
i = 7
i == 7
i > 15
i >= 15
i <= 25
i >= 15 && i <= 25
i = 30
i >= 15 && i <= 25
i = 5
i >= 15 && i <= 25
true && true
true && false
false && true
false && false

false || true
false || true
```

```
true || false
true || true
false || false
false ^ false false ^ true
true ^ false
true ^ true !true
!false int x = 6;
!(x>7)

!(x>7)
true || ++i==11
i
int i = 10;
int j = 15;
j > 15 && i++ > 5
j
i
j > 15 & i++ > 5
j
i
i++;
i++;
char ch2 = '\u0022';
char ch3 = '\u00A2';
ch++
ch
++ch
++ch
ch + 5
ch
(int)ch
ch
System.out.println(ch);

char ch = '\t';
System.out.println(ch);
```

```
System.out.println(ch);
(int)'1'
(int)'0' (int)'9'
(int)'2'
(int)'a' (int)'z'
(int)'A' (int)'Z'
                for (char ch = 'A'; ch <= 'Z'; ch++) {
                        System.out.println(ch);

}
                for (char ch = 'A'; ch <= 'Z'; ch++) {
                        System.out.println(ch);
}
                for (char ch = 'A'; ch <= 'C'; ch++) {
                        System.out.println(ch);
}
```

/06-PrimitiveDataTypesAndAlternatives/src/com/in28minutes/primitive/datatypes/BiNumber.java

```
package com.in28minutes.primitive.datatypes;

public class BiNumber {
        private int number1;
        private int number2;

        public int getNumber1() {
                return number1;
        }

        public void setNumber1(int number1) {
                this.number1 = number1;

        }
```

```
        public int getNumber2()  {
                return number2;
        }

        public void setNumber2(int number2)  {
                this.number2 = number2;
        }

        public BiNumber(int number1, int number2)

 {
                this.number1 = number1;
                this.number2 = number2;
        }

        public int add() {
                return number1 + number2;
        }

        public int multiply() {
                return number1 * number2;
        }

        public void doubleValue() {
                this.number1 *= 2;
                this.number2 *= 2;
        }
}
```

/06-PrimitiveDataTypesAndAlternatives/src/com/in28minutes/primitive/datatypes/BiNumberRunner.java

```
package com.in28minutes.primitive.datatypes;

public class BiNumberRunner {
```

```
        public static void main(String[] args)
 {
                BiNumber numbers = new BiNumber(2,  3);
                System.out.println(numbers.add());//2+3

        System.out.println(numbers.multiply());//2*3
                numbers.doubleValue();//Double both numbers

System.out.println(numbers.getNumber1());//4
System.out.println(numbers.getNumber2());//6
        }
}
```

/06-PrimitiveDataTypesAndAlternatives/src/com/in28minutes/primitive/datatypes/MyChar.java

```
package com.in28minutes.primitive.datatypes;

public class MyChar {

        private char ch;

        public MyChar(char ch)
 {
                this.ch = ch;
        }
```

```
	public boolean isVowel() {
		//'a' e i o u or A E I O
U
		if(ch == 'a' || ch == 'A')
			return
true;
		if(ch == 'e' || ch ==
'E')
			return true;
		if(ch == 'i' || ch == 'E')
			return true;
		if(ch == 'o' || ch == 'O')
			return true;
		if(ch == 'u' || ch == 'U')
			return true;
		return false;
	}
	public boolean isDigit() {
		if(ch >= 48 && ch <=57) //between '0' and
'9'
		  return true;
		return false;
	}
	public boolean isAlphabet() {
		if(ch >= 97 && ch <=122) //between 'a' and
```

```
 'z'
                  return  true;

                if(ch >= 65 && ch <=90) //between 'A' and
 'Z'
                        return  true;

                return false;
        }


        public boolean isConsonant() {
                //Alphabet and it is not VOWEL
                //! [a , e, i ,o , u]
                if(isAlphabet() && !isVowel())
                        return true;

                return false;

        }

        public static void printLowerCaseAlphabets() {
                //'a' to 'z'
                for (char ch = 'a'; ch <= 'z'; ch++) {
                        System.out.println(ch);
                }
        }

        public static void printUpperCaseAlphabets() {
                for (char ch = 'A'; ch <= 'Z'; ch++) {
                        System.out.println(ch);
                }

        }
}
```

/06-PrimitiveDataTypesAndAlternatives/src/com/in28minutes/primitive/datatypes/MyCharRunner.java

```
package com.in28minutes.primitive.datatypes;
 public class MyCharRunner {

        public static void main(String[] args) {
                MyChar myChar = new  MyChar('B');



        System.out.println(myChar.isVowel());

                //'a', 'e', 'i', 'o', 'u' and Capitals
                System.out.println(myChar.isDigit());
                System.out.println(myChar.isAlphabet());
//'a' to 'z' or 'A' to 'Z'

                System.out.println(myChar.isConsonant());

                MyChar.printLowerCaseAlphabets();
                MyChar.printUpperCaseAlphabets();
        }

}
```

/06-PrimitiveDataTypesAndAlternatives/src/com/in28minutes/primitive/datatypes/SimpleInterestCalculator.java

```
package com.in28minutes.primitive.datatypes;

import java.math.BigDecimal;

public class SimpleInterestCalculator
```

```
 {

        BigDecimal principal;

        BigDecimal interest;


        public SimpleInterestCalculator(String principal,
String interest) {
                this.principal = new BigDecimal(principal);
                this.interest = new
BigDecimal(interest).divide(new BigDecimal(100));
        }



        public BigDecimal calculateTotalValue(int
noOfYears) {
                // Total Value = principal + principal *
interest * noOfYears;
                BigDecimal noOfYearsBigDecimal = new
BigDecimal(noOfYears);
                BigDecimal totalValue =
principal.add(principal.multiply(interest).multiply(noOfYea
rsBigDecimal));
                return totalValue;
        }

}
```

/06-PrimitiveDataTypesAndAlternatives/src/com/in28minutes/primitive/datatypes/SimpleInterestCalculatorRunner.java

```
package com.in28minutes.primitive.datatypes;

import java.math.BigDecimal;
```

```
public class SimpleInterestCalculatorRunner {

        public static void main(String[] args) {

                SimpleInterestCalculator calculator =
                        new
SimpleInterestCalculator("4500.00",

 "7.5");
                 BigDecimal totalValue =

calculator.calculateTotalValue(5);// 5 years
                  //6187.50000
                 System.out.println(totalValue);
        }

}
```

Conditionals - If, Switch and More..

Steps

- Step 00 - Conditionals with Java - Section Overview
- Step 01 - Introduction to If Else Statement
- Step 02 - Introduction to Nested If Else
- Step 03 - If Else Statement - Puzzles
- Step 04 - If Else Problem - How to get User Input in Java?
- Step 05 - If Else Problem - How to get number 2 and choice from user?
- Step 06 - If Else Problem - Implementing with Nested If Else
- Step 07 - Java Switch Statement - An introduction
- Step 08 - Java Switch Statement - Puzzles - Default, Break and Fall Through
- Step 09 - Java Switch Statement - Exercises - isWeekDay, nameOfMonth, nameOfDay
- Step 10 - Java Ternary Operation - An Introduction
- Step 11 - Conditionals with Java - Conclusion

Exercises

If and Nested If Else - Design a Menu

- Ask User for input
  - Enter two numbers
  - Choose an Operation
    - add
    - multiply
    - divide
    - subtract

- ◦ ◦ ...
    - Publish Result

```
Enter Number1:
2

Enter Number2:

4

1 - Add
2 - Subtract
3 - Divide
4 - Multiply
Choose Operation: 4

Result is - 8
```

Switch

- `public static boolean isWeekDay(int dayNumber) {`
    - input - number of day 0 (Sunday) to 6(Saturday)
    - return if the day is a Week Day.
- `public static String determineNameOfMonth(int monthNumber) {`
    - input - number of month 1(January) to 12(December)
    - output - Name of month
- `public static String determineNameOfDay(int dayNumber) {`
    - input - number of day 0 (Sunday) to 6(Saturday)
    - Return the day of week in text

Code Examples

/07-Conditionals/commands.txt

```
if(true)
```

```
 {
   System.out.println("True");

}
if(false) {
   System.out.println("True");
}
if(i==3) {
   System.out.println("True");
}


if(i<2) {
    System.out.println("True");
}
if(i<=3 || i>=35) {
    System.out.println("True");
}
if(i<=3 && i>=35) {
    System.out.println("True");
}
if (i==3) {
   System.out.println("True");
} else {
   System.out.println("i is not 3");
}
i = 5
if (i==3) {
   System.out.println("True");
} else {
   System.out.println("i is not 3");
}
if(i==1) {
   System.out.println("i");
}
switch (i) {
   case 1 :
```

```
 System.out.println("1");
   case 5 : System.out.println("5");
   default : System.out.println("default");
}
i = 1 switch (i)

 {
   case 1 : System.out.println("1");
   case 5 : System.out.println("5");
   default : System.out.println("default");
} switch (i)


 {
   case 1 : System.out.println("1"); break;
   case 5 : System.out.println("5"); break;
   default : System.out.println("default"); break;
}
boolean isEven;
int i =5;
if(i%2==0) {
  isEven = true;
} else {
  isEven = false;
}
isEven
i = 6
if(i%2==0) {
  isEven = false;
}
if(i%2==0) {
  isEven = true;
} else {
  isEven = false;
}
isEven
isEven = ( i%2==0 ? true : false  )
```

```
i = 6
isEven = ( i%2==0 ? true : false  )
i = 7
isEven = ( i%2==0 ? true : false  )
i = 6
String even = ( i%2 ==0 ? "YES" : "NO" );
```

/07-Conditionals/src/com/in28minutes/ifstatement/examples/IfStatementRunner.java

```
package com.in28minutes.ifstatement.examples;

public class IfStatementRunner {

	public static void main(String[] args) {
		puzzle5();
	}

	private static void puzzle1() {
		int k = 15;
		if (k > 20) {
			System.out.println(1);
		} else if (k > 10) {
			System.out.println(2);
		} else if (k < 20) {
			System.out.println(3);
		} else {
			System.out.println(4);
		}

	}

	private static void puzzle2() {
		int l =
```

```
15;

                if (l < 20)
                        System.out.println("l<20");//
                if (l > 20)
                        System.out.println("l>20");
                else
                        System.out.println("Who am I?");//

        }



        private static void puzzle3()

{
                int m =

15;

                if(m>20)
                    if(m<20)
                System.out.println("m>20");
                    else
                System.out.println("Who am I?");
        }


        private static void puzzle5() {
                int number = 5;
                if(number < 0)
                    number = number + 10;
                number++;
                System.out.println(number);
        }

        private static void basicNestedIfElse()
```

```
 {
                int i = 24;
                // i is 25
                // i is 24
                // i is neither 25 or 24
                if (i == 25) {
                        System.out.println("i = 25");
                } else if (i == 24) {
                        System.out.println("i = 24");
                } else if (i == 23) {
                        System.out.println("i = 23");
                } else {
                        System.out.println("i != 24 and i

 !=25 and i !=23");
                }

        }

}
```

/07-Conditionals/src/com/in28minutes/ifstatement/examples/MenuRunner.java

```
package com.in28minutes.ifstatement.examples;

import java.util.Scanner;

public class MenuRunner {
        public static void main(String[] args) {
                // Type obj = new Type(argument);
                Scanner scanner = new Scanner(System.in);
                System.out.print("Enter Number1: ");
                int number1 = scanner.nextInt();

                System.out.print("Enter Number2:
```

```
 ");
                int number2 = scanner.nextInt();

                System.out.println("Choices Available are
");
                System.out.println("1 - Add");
                System.out.println("2 - Subtract");
                System.out.println("3 - Divide");
                System.out.println("4 - Multiply");

                System.out.print("Enter Choice: ");
                int choice = scanner.nextInt();

                System.out.println("Your Choices are");
                System.out.println("Number1 " +

 number1);
                System.out.println("Number2 " +

 number2);
                System.out.println("Choice " +

 choice);

                performOperationUsingSwitch(number1,
number2, choice);
        }

        private static void
performOperationUsingNestedIfElse(int number1, int number2,
int choice) {
                if (choice == 1) {
                        System.out.println("Result " +
(number1 + number2));
                } else if (choice == 2) {
                        System.out.println("Result " +
(number1 - number2));
                } else if (choice == 3)
```

```
 {
                        System.out.println("Result " +
(number1 / number2));
                } else if (choice == 4) {
                        System.out.println("Result " +
(number1 * number2));
                } else {
                        System.out.println("Invalid
Operation");
                }
        }

        private static void performOperationUsingSwitch(int
number1, int number2, int choice) {
                switch (choice) {
                case  1:
                        System.out.println("Result " +
(number1 + number2));


        break;
                case

 2:
                        System.out.println("Result " +
(number1 - number2));
                        break;
                case 3:
                        System.out.println("Result " +
(number1 / number2));
                        break;
                case 4:
                        System.out.println("Result " +
(number1 * number2));
                        break;
                default:
                        System.out.println("Invalid
```

```
 Operation");
                        break;
                }

        }

}
```

/07-Conditionals/src/com/in28minutes/ifstatement/examples/SwitchExercisesRunner.java

```
package com.in28minutes.ifstatement.examples;

public class SwitchExercisesRunner {

        public static void main(String[] args) {
                System.out.println(isWeekDay(5));
        }


        public static boolean isWeekDay(int dayNumber) {
                switch(dayNumber)

 {
                //case 0 :
                //case 6 : return false;
                case 1 :
                case 2 :
                case 3 :
                case 4 :
                case 5 : return true;
                }

                return false;
```

```
        }

        public static String determineNameOfDay(int
dayNumber) {
                switch (dayNumber) {
                case 0:
                        return "Sunday";
                case 1:
                        return "Monday";
                case 2:
                        return "Tuesday";
                case 3:
                        return "Wednesday";
                case 4:
                        return "Thursday";
                case 5:
                        return  "Friday";
                case 6:
                        return "Saturday";
                }

                return "Invalid_day";

        }
}
```

/07-Conditionals/src/com/in28minutes/ifstatement/examples/SwitchStatementRunner.java

```
package com.in28minutes.ifstatement.examples;

public class SwitchStatementRunner {
        public static void main(String[] args) {
                puzzle4();
        }
```

```
	private static void puzzle1() {
		int number = 2;
		switch (number) {
		case 1:
			System.out.println(1);
		case 2:
			System.out.println(2);
		case 3:
			System.out.println(3);
		default:
			System.out.println("Default");
		}
	}

	private static void puzzle2()  {
		int number = 2;
		switch (number) {
		case 1:
			System.out.println(1);
			break;
		case 2:
		case
 3:
			System.out.println("Number is 2 or
3");


	break;
		default:
			System.out.println("Default");
			break;
		}
	}

	private static void puzzle3()
```

```
{
		int number = 10;
		switch (number) {
		case 1:
			System.out.println(1);
			break;
		case 2:
			System.out.println(2);
			break;
		case 3:
			System.out.println(3);
			break;
		default:
			System.out.println("Default");
			break;
		}
	}

	private static void puzzle4() {
		int number = 10;
		switch (number) {
		default:
			System.out.println("Default");


	break;
		case 1:
			System.out.println(1);
			break;
		case

2:


	System.out.println(2);
			break;
		case
```

```
 3:
                        System.out.println(3);
                        break;
                }
        }

        private static void puzzle5() {
                long l = 15;
                /*switch(l){

                }*/
        }

        private static void puzzle6() {
                int number =  10;
                int i = number * 2;
                switch (number) {
                 //case number>5:
System.out.println("number>5");
                }
        }

}
```

Loops

Steps

- Step 00 - Java Loops - Section Introduction
- Step 01 - Java For Loop - Syntax and Puzzles
- Step 02 - Java For Loop - Exercises Overview and First Exercise Prime Numbers
- Step 03 - Java For Loop - Exercise - Sum Upto N Numbers and Sum of Divisors
- Step 04 - Java For Loop - Exercise - Print a Number Triangle
- Step 05 - While Loop in Java - An Introduction
- Step 06 - While Loop - Exericises - Cubes and Squares upto limit
- Step 07 - Do While Loop in Java - An Introduction
- Step 08 - Do While Loop in Java - An Example - Cube while user enters positive numbers
- Step 09 - Introduction to Break and Continue
- Step 10 - Selecting Loop in Java - For vs While vs Do While

Exercises

For Loop
Implement MyNumber class with behavior shown in the example below:

```
MyNumber number = new MyNumber(9);

number.isPrime(); //Is a number Prime?
//Hint : 5 => true, 7 => true, 11 => true, 6 => false

int sum = number.sumUptoN();//Sum of numbers upto n?
//1 + 2 + 3 + 4 + 5 + 6

int sumOfDivisors = number.sumOfDivisors();



number.printANumberTriangle();
//1
//1 2
//1 2 3
```

```
//1 2 3 4  //1 2 3 4 5
```

While

Implement WhileNumberPlayer class with behavior shown in the example below:

```
WhileNumberPlayer player = new
WhileNumberPlayer(30);//limit

player.printSquaresUptoLimit();
//For limit = 30, output would be 1 4 9 16 25

player.printCubesUptoLimit();
//For limit = 30, output would be 1 8 27
```

Choosing Loops

Thinking Exercise

- What would we use for the Menu
- If we would want to run the Menu again and again?

```
Enter Number1:
2

Enter Number2:
4

1 - Add
2 - Subtract
3 - Divide
4 - Multiply
5 - Exit

Choose Operation: 4

Result is 8

Choose Operation: 1
Result is
```

```
 6
  Choose Operation: 5
 Thank You!
```

Code Examples

/08-Loops/commands.txt

```
for (int i = 0; i<= 10; i++) {
   System.out.print (i + " ");
}
for (int i = 0; i<= 10; i = i + 2) {
   System.out.print (i + " ");
}
for (int i = 1; i<= 10; i = i + 2) {
   System.out.print (i + " ");
}
for (int i = 11; i<= 10; i = i + 2) {
   System.out.print (i + " ");
}
for (int i = 11; i<= 20;) {
   System.out.print (i + " ");
   i++;
}
for (; i<= 30;i++) {
   System.out.print (i + " ");
}
9 % 2
9 % 3

if (i>2)


 {
   System.out.println("i>2");
```

```
}
int i = 3; if (i>2)  {
   System.out.println("i>2"); }
i = 0
while (i <5) {
   System.out.println(i);
   i++;
}
i
i = 6
while (i < 5) {
   System.out.println(i);
   i++;
}
i = -2
while (i < 5) {
   System.out.println(i);
}
while (i < 5) {
   System.out.println(i);
   i++;
}
i
while (i < 5) {
   System.out.print(i + " ");
   i++;
}
i
i = 1
do {
   System.out.print(i + "

 ");
   i++;

} while (i<5);
i = 10
```

```
while (i < 5)   {
   System.out.print(i + " ");
   i++; }
do {
   System.out.print(i + " ");
   i++;
} while (i<5);
for(i=1;i<=10;i++) {
   if(i==5)
      break;
   System.out.print(i + " ");
}
for(i=1;i<=10;i++) {
   if(i%2==0)
      break;
   System.out.print(i + " ");
}
for(i=1;i<=10;i++) {
   if(i%2==0)
     continue;
   System.out.print(i + " ");
}
for(i=1;i<=10;i++) {
   if(i%2!=0)
     continue;
   System.out.print(i + " ");
}
```

/08-Loops/src/com/in28minutes/loops/DoWhileRepeatedQuestionRunner.java

```
package com.in28minutes.loops;

import java.util.Scanner;
 public class DoWhileRepeatedQuestionRunner
```

```
 {

	public static void main(String[] args) {

		Scanner scanner = new Scanner(System.in);
		int number = -1;

		do {
			if (number != -1) {
				System.out.println("Cube is
" + (number * number * number));
			}
			System.out.print("Enter a number:
");
			number = scanner.nextInt();
		} while (number >= 0);
		System.out.print("Thank You! Have Fun!");
	}

}
```

/08-Loops/src/com/in28minutes/loops/MyNumber.java

```
package com.in28minutes.loops;

public class MyNumber {

	private int number;

	public MyNumber(int number) {
		this.number = number;
	}


	public boolean isPrime()
```

```
 {
                // 2 to number-1
                // How can check if a number is divisible
by 2?

                if (number < 2) {
                        return false;
                }

                for (int i = 2; i <= number - 1; i++) {
                        if (number % i == 0) {
                                return false;
                        }
                }

                return true;
        }

        public int sumUptoN() {
                int sum = 0;

                for (int i = 1; i <= number; i++) {
                        sum = sum + i;
                }

                return sum;
        }

        public int sumOfDivisors() {
                // 6 except 1 , 6 => 2,3
                // 2 + 3 + 4 + 5

                int sum =
```

```
0;

                for (int i = 2; i <= number - 1; i++) {
                        if (number % i == 0)

{
                                sum = sum +  i;
                        }
                }

                return sum;
        }

        public void printNumberTriangle() {
                // 1
                // 1 2
                // 1 2 3
                // 1 2 3 4
                // 1 2 3 4 5

                for (int i = 1; i <= number; i++) {
                        for (int j = 1; j <= i; j++) {
                                System.out.print(j + " ");
                        }
                        System.out.println();
                }

        }

}
```

/08-Loops/src/com/in28minutes/loops/MyNumberRunner.java

```
package com.in28minutes.loops;

import com.in28minutes.loops.MyNumber;
```

```
public class MyNumberRunner {

	public static void main(String[] args)

 {
		MyNumber number = new MyNumber(5);

		boolean isPrime = number.isPrime();
		System.out.println("isPrime " + isPrime);

		int sum = number.sumUptoN();
		System.out.println("sumUptoN " + sum);

		int sumOfDivisors = number.sumOfDivisors();
		System.out.println("sumOfDivisors " +
sumOfDivisors);

		number.printNumberTriangle();
	}

}
```

/08-Loops/src/com/in28minutes/loops/WhileNumberPlayer.java

```
package com.in28minutes.loops;

public class WhileNumberPlayer {

	private int limit;

	public WhileNumberPlayer(int limit) {
		this.limit = limit;
	}
```

```
        // For limit = 30, output would be 1 4 9 16 25

        public void printSquaresUptoLimit() {
                int i = 1;
                while (i * i < limit) {
                        System.out.print(i * i + "

 ");

                        i++;
                }
                System.out.println();
        }

        // For limit = 27, output would be 1 8 27
        public void printCubesUptoLimit() {
                int i = 1;
                while (i * i * i <= limit) {
                        System.out.print(i * i * i + " ");
                        i++;
                }
                System.out.println();
        }

}
```

/08-Loops/src/com/in28minutes/loops/WhileNumberPlayerRunner.java

```
package com.in28minutes.loops;

public class WhileNumberPlayerRunner {
        public static void main(String[] args) {
                WhileNumberPlayer player = new
WhileNumberPlayer(27);
```

```
        player.printSquaresUptoLimit();
                player.printCubesUptoLimit();


        }

}
```

Reference Types

Steps

- Step 00 - Java Reference Types - Section Introduction
- Step 01 - Reference Types - How are they stored in Memory?
- Step 02 - Java Reference Types - Puzzles
- Step 03 - String class - Introduction and Exercise - Print each word and char on a new line
- Step 04 - String class - Exercise Solution and Some More Important Methods
- Step 05 - Understanding String is Immutable and String Concat, Upper Case, Lower Case, Trim methods
- Step 06 - String Concatenation and Join, Replace Methods
- Step 07 - Java String Alternatives - StringBuffer and StringBuilder
- Step 08 - Java Wrapper Classes - An Introduction - Why and What?
- Step 09 - Java Wrapper Classes - Creation - Constructor and valueOf
- Step 10 - Java Wrapper Classes - Auto Boxing and a Few Wrapper Constants - SIZE, BYTES, MAX_VALUE and MIN_VALUE
- Step 11 - Java Dates - Introduction to LocalDate, LocalTime and LocalDateTime
- Step 12 - Java Dates - Exploring LocalDate - Creation and Methods to play with Date
- Step 13 - Java Dates - Exploring LocalDate - Comparing Dates and Creating Specific Dates
- Step 14 - Java Reference Types - Conclusion

Exercises

String

- Take a piece of Text into a String.
  - Print each character in the text on a seperate line
  - Print each word in the text on a seperate line

Code Examples

/09-ReferenceTypes/commands.txt

```
class Planet {
}

Planet jupiter = new Planet();
int i = 5;
class Animal {
    int id;
    Animal(int id) {
       this.id = id;
    }
}
Animal nothing;
nothing = cat
nothing.id = 10
cat.id
nothing = dog
nothing.id
int j = i;
j = 6
i
i == j
j = 5
i == j
Animal dog = new Animal(12);
Animal cat = new Animal(10);
Animal ref = cat;
Animal dog2 = new Animal(12);
cat == dog
cat == ref
dog == dog2
1
2
```

```
12.34

"Test".length()
BigDecimal bd = new BigDecimal("1.0");
str.charAt(0)
str.charAt(2) str.charAt(3)

String biggerString = "This is a lot of text";
str.substring(5)
biggerString.substring(5)
biggerString.substring(5,13)
str.charAt(13)
biggerString.charAt(13)
biggerString.charAt(456)
someString.length()
someString.charAt(5)
for(int i= 0; i<someString.length(); i++) {
    System.out.println(someString.charAt(i));
}
someString.indexOf("lot")
someString.charAt(10)
someString.charAt('i')
someString.indexOf('i')
someString.lastIndexOf('i')
someString.contains("text")
String someString = "This is a lot of text again";
someString.startsWith("This")
someString.startsWith("jfsdklfj")
someString.endsWith("jfsdklfj")
someString.endsWith("in")
someString.endsWith("ain")
someString.endsWith("again")
someString.endsWith("againfsda")
someString.isEmpty()
"fjsadlkfj".isEmpty()
```

```
"".isEmpty()

"true".equals("true")
"value".equals("value")

str.equals("value") str.equals("VAlue")
str.equalsIgnoreCase("VAlue") str.concat("is awesome");

str
String anotherString = str.concat(" is awesome");
str
anotherString
String string2 = anotherString.concat(".");
str
anotherString
string2
str.toUpperCase()
str.toLowerCase()
str2.trim()
String str2 = "  in28Minutes is awesome.   ";
str2.trim()
1 + 2
"1" + "2"
"1" + 2
"1" + 23
1 + 23
1 + 2 + "3"
"1" + 2 + 3
System.out.println("Value is " + 20)
System.out.println("Value is " + 20 + 20)
System.out.println("Value is " + (20 + 20))
String.join(",", "2", "3", "4")
String.join(",", "A", "B","C")
String.join(",", "A")
String.join(",", "A", "B")
"abcd".replace('a', 'z');
```

```
"abcd".replace("ab", "xyz");

String str = "jdsfja ";
"123" + "123" + "1234" + "123456"
sb.append(" 123")

sb sb
sb.setCharAt(1,'e')

sb
StringBuilder sb = new StringBuilder("test");
Integer integer1 = new Integer("5234");
Integer integer2 = new Integer("5234");
Integer i1 = new Integer(5);
Integer i2 = new Integer(5);
Integer i3 = Integer.valueOf(5);
Integer i4 = Integer.valueOf(5);
i1 == i2
i3 == i4
Integer integer = Integer.valueOf("4567");
int i = integer.intValue();
Float f = Float.valueOf("12.45");
f.floatValue()
f.intValue()
Integer eight = Integer.valueOf(8);
Integer.toBinaryString(eight);
Integer.toHexString(eight);
Integer eightyEight = Integer.valueOf(88);
Integer.toHexString(eightyEight);
seven++
seven
seven == sevenAgain
Integer seven = 7;
Integer sevenAgain = 7;
seven == sevenAgain
Integer.MAX_VALUE
Integer.MIN_VALUE
```

```
Integer.SIZE

Integer.BYTES
import java.time.LocalDate;
import java.time.LocalDateTime; import java.time.*;

today.getYear()
today.getDayOfWeek()

today.getDayOfMonth()
today.getDayOfYear()
today.getMonth()
today.getMonthValue()
today.isLeapYear()
today.lengthOfYear()
today.lengthOfMonth()
today.plusDays(100)
today.plusMonths(100)
today.plusYears(100)
today.minusYears(100)
LocalDate hundredYearsBefore = today.minusYears(100);
today
LocalDateTime now = LocalDateTime.now();
LocalDate today = LocalDate.now();
LocalDate yesterady = LocalDate.of(2018, 01, 31);
LocalDate yesterday = LocalDate.of(2018, 01, 31);
today
yesterday
today.withYear(2016)
today.withDayOfMonth(20)
today.withMonth(3)
today.withDayOfYear(120)
today.isBefore(yesterday)
today.isAfter(yesterday)
```

Arrays And ArrayList

Steps

- Step 00 - Introduction to Array and ArrayList - Section Introduction with a Challenge
- Step 01 - Understanding the need and Basics about an Array
- Step 02 - Java Arrays - Creating and Accessing Values - Introduction
- Step 03 - Java Arrays - Puzzles - Arrays of Objects, Primitive Data Types, toString and Exceptions
- Step 04 - Java Arrays - Compare, Sort and Fill
- Step 05 - Java Arrays - Exercise - Create Student Class - Part 1 - Total and Average Marks
- Step 06 - Java Arrays - Exercise - Create Student Class - Part 2 - Maximum and Minimum Mark
- Step 07 - Introduction to Variable Arguments - Need
- Step 08 - Introduction to Variable Arguments - Basics
- Step 09 - Introduction to Variable Arguments - Enhancing Student Class
- Step 10 - Java Arrays - Using Person Objects and String Elements with Exercises
- Step 11 - Java String Arrays - Exercise Solutions - Print Day of Week with Most number of letters and more
- Step 12 - Adding and Removing Marks - Problem with Arrays
- Step 13 - First Look at ArrayList - An Introduction
- Step 14 - First Look at ArrayList - Refactoring Student Class to use ArrayList
- Step 15 - First Look at ArrayList - Enhancing Student Class with Add and Remove Marks
- Step 16 - Introduction to Array and ArrayList - Conclusion

Exercises

Student Class

```
Student student = new Student (name, list of marks);
int number = student.getNumberOfMarks();
int sum = student.getTotalSumOfMarks();
int maximumMark = student.getMaximumMark();
int minimumMark = student.getMinimumMark();
BigDecimal average = student.getAverageMarks();



 student.addNewMark(35);
student.removeMarkAtIndex(5);
```

String Arrays

- Create a string array with days of the week
  - "Sunday", "Monday", "Tuesday", "Wednesday"
  - "Thursday", "Friday", "Saturday"
- Find the day with most number of letters in it
  - Longest String
- Print days of the week backwards

Code Examples

/10-ArraysAndArrayList/src/com/in28minutes/arrays/StringRunner.java

```
package com.in28minutes.arrays;

public class StringRunner {

        public static void main(String[] args) {

                String[] daysOfWeek = { "Sunday", "Monday",
"Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
```

```
 };

                String dayWithMostCharacters =



 "";

                for (String day : daysOfWeek)

 {
                        if (day.length() >
dayWithMostCharacters.length())   {
                                dayWithMostCharacters =

 day;
                        }
                }

                System.out.println("Day with Most number of
characters " + dayWithMostCharacters);

                for (int i = daysOfWeek.length - 1; i >= 0;
i--) {
                        System.out.println(daysOfWeek[i]);
                }


        }

}
```

/10-ArraysAndArrayList/src/com/in28minutes/arrays/Student.java

```
package com.in28minutes.arrays;

import java.math.BigDecimal;
```

```
import java.math.RoundingMode;
import java.util.ArrayList;
import java.util.Collections;

public class Student {




        private String name;

        private ArrayList<Integer> marks = new



 ArrayList<Integer>();

        public Student(String name, int... marks)

 {
                this.name =    name;

                for (int mark : marks) {
                        this.marks.add(mark);
                }
        }

        public int getNumberOfMarks() {
                return marks.size();
        }

        public int getTotalSumOfMarks() {
                int sum = 0;
                for (int mark : marks) {
                        sum += mark;
```

```
        }
                return sum;

        }

        public int getMaximumMark() {
                return Collections.max(marks);

        }

        public int getMinimumMark() {
                return Collections.min(marks);


        }



        public BigDecimal getAverageMarks()

{
                int sum = getTotalSumOfMarks();
                int number =

getNumberOfMarks();

                return new BigDecimal(sum).divide(new

BigDecimal(number), 3, RoundingMode.UP);

        }

        public String toString() {
                return name + marks;
        }

        public void addNewMark(int mark) {
                marks.add(mark);
        }
```

```
        public void removeMarkAtIndex(int index) {

        marks.remove(index);

        }
}
```

/10-ArraysAndArrayList/src/com/in28minutes/arrays/StudentRunner.java

```
package com.in28minutes.arrays;

import java.math.BigDecimal;

public class StudentRunner {


        public static void main(String[] args) {

                Student student = new Student("Ranga", 97,
98,

 100);

                int number = student.getNumberOfMarks();
                System.out.println("number of marks : " +

 number);

                int sum =

 student.getTotalSumOfMarks();
                System.out.println("sum of marks : " +
```

```
 sum);

                int maximumMark = student.getMaximumMark();
                System.out.println("maximum of marks : " +

 maximumMark);

                int minimumMark =

 student.getMinimumMark();
                System.out.println("minimum of marks : " +
minimumMark);

                BigDecimal average =
student.getAverageMarks();
                System.out.println("average : " + average);



        System.out.println(student);

                student.addNewMark(35);



        System.out.println(student);

                student.removeMarkAtIndex(1);

                System.out.println(student);

        }
}
```

Object Oriented Programming Again

Steps

- Step 00 - Object Oriented Programming - Level 2 - Section Introduction
- Step 01 - Basics of Designing a Class - Class, Object, State and Behavior
- Step 02 - OOPS Example - Fan Class - Deciding State and Constructors
- Step 03 - OOPS Example - Fan Class - Deciding Behavior with Methods
- Step 04 - OOPS Exercise - Rectangle Class
- Step 05 - Understanding Object Composition with Customer Address Example
- Step 06 - Understanding Object Composition - An Exercise - Books and Reviews
- Step 07 - Understanding Inheritance - Why do we need it?
- Step 08 - Object is at top of Inheritance Hierarchy
- Step 09 - Inheritance and Overriding - with toString() method
- Step 10 - Java Inheritance - Exercise - Student and Employee Classes
- Step 11 - Java Inheritance - Default Constructors and super() method call
- Step 12 - Java Inheritance - Puzzles - Multiple Inheritance, Reference Variables and instanceof
- Step 13 - Java Abstract Class - Introductio

- Step 14 - Java Abstract Class - First Example - Creating Recipes with Template Method
- Step 15 - Java Abstract Class - Puzzles
- Step 16 - Java Interface - Example 1 - Gaming Console - How to think about Intefaces?
- Step 17 - Java Interface - Example 2 - Complex Algorithm - API defined by external team
- Step 18 - Java Interface - Puzzles - Unimplemented methods, Abstract Classes, Variables, Default Methods and more
- Step 19 - Java Interface vs Abstract Class - A Comparison
- Step 20 - Java Interface Flyable and Abstract Class Animal - An Exercise
- Step 21 - Polymorphism - An introduction

Exercises

Creating a simple class

- public class Rectangle
  - length, width;
  - What constructors?
  - What Operations?

Object Composition - Book and Reviews

Book > id, name, author

Reviews > id, description, rating

```
Book book =
   new Book(123, "Object Oriented Programming with Java",
         "Ranga");
```

```
book.addReview(
   new Review(10, "Great Book", 5));
book.addReview(
   new Review(101, "Awesome", 5);

System.out.println(book);
```

Inheritance

Setup an Inheritance Hierarchy and implement toString in Employee class

- Person
  - name,phone,email;
- Student
  - college, class
- Employee
  - title, employer, employeeGrade, salary
  - toString (print all values including those of Person)

Interface and Abstract Class

interface Flyable

- void fly();
- Bird "with wings"
- Aeroplane "with fuel"
- Flyable flyingObjects = {new Bird(), new Aeroplane()};
- Loop and invoke fly method

abstract class Animal

- void bark()
- Dog "Bow Bow"
- Cat "Meow Meow"
- Animal[] animals = {new Cat(), new Dog()};

- Loop and invoke bark method

Code Examples

/commands.txt

```
class Pet extends Animal {
     public void groom()   {
        System.out.println("Groom");
     } }
dog.toString()
dog.groom()
Pet pet = new Dog();
pet.groom()
pet instanceof Pet
pet instanceof Dog
pet instanceof Animal
pet instanceof Object
animal instanceof Pet
animal instanceof Dog
animal instanceof Object
class Animal {
   public void bark() {
       System.out.println("TEst");
   }
}
animal.bark()
abstract class AbstractAnimal {
    abstract public void bark();
}
class Dog extends AbstractAnimal {
    public void bark() {
       System.out.println("Bow Bow");
    }
}
Dog dog = new Dog();
```

```
dog.bark()
abstract class AbstractTest {
}
abstract class Algorithm1 extends AbstractAlgorithm {
}
abstract class AbstractAlgorithm   {
     private int stepCount;
     public int getStepCount()  {
          return stepCount();
     }
}
class Implementation implements Interface2 {
   public void method2() { }
   public void method1() { }
}
abstract class ImplementationAbstract implements Interface2
{
   public void method1() { }
}
interface Interface3 {
  int test = 5;
}
interface Interface4 {
    default void print() {
       System.out.println("default");
    }
}
class Test implements Interface4 {
}
test.print()
class Test1 implements Interface4 {
   public void print() {
      System.out.println("override");
   }
}
```

```
Test1 test = new Test1(); test.print()

interface Interface1 {
 void method1();
}
interface Interface2  {
  void method2();
}
```

/src/com/in28minutes/oops/level2/AbstractRecipe.java

```
package com.in28minutes.oops.level2;

public abstract class AbstractRecipe {

        public void execute() {
                getReady();
                doTheDish();
                cleanup();
        }

        abstract void getReady();
        abstract void doTheDish();
        abstract void cleanup();
}
```

/src/com/in28minutes/oops/level2/Address.java

```
package com.in28minutes.oops.level2;

public class Address {
        private String line1;
        private String city;
        private String zip;
```

```
        //creation
        public Address(String line1, String city, String
 zip) {
                super();
                this.line1 =  line1;
                this.city = city;
                this.zip = zip;
        }

        public String toString() {
                return line1 + " " + city + " " + zip;

        }

}
```

/src/com/in28minutes/oops/level2/Book.java

```
package com.in28minutes.oops.level2;

import java.util.ArrayList;

public class Book {

        private int id;
        private String name;
        private String author;
        private ArrayList<Review> reviews = new ArrayList<>
();

        public Book(int id, String name, String author) {
                this.id = id;
                this.name = name;
                this.author = author;
        }
```

```
        public void addReview(Review review)  {
                this.reviews.add(review);


        }

        public String toString() {
                return String.format("id =%d name = %s
author = %s Reviews = [%s]",
                                id, name, author, reviews);
        }

}
```

/src/com/in28minutes/oops/level2/BookRunner.java

```
package com.in28minutes.oops.level2;

public class BookRunner {

        public static void main(String[] args) {
                Book book = new Book(123, "Object Oriented
Programming with Java",
                                "Ranga");
                book.addReview(new Review(10, "Great Book",
5));
                book.addReview(new Review(101, "Awesome",
5));

                System.out.println(book);

        }

}
```

/src/com/in28minutes/oops/level2/Customer.java

```
package com.in28minutes.oops.level2;

public class Customer {

	//state
	private String name;
	private Address homeAddress;
	private Address workAddress;

	//creating
	public Customer(String name, Address homeAddress) {
		this.name = name;
		this.homeAddress = homeAddress;
	}

	//operations
	public Address getHomeAddress() {
		return homeAddress;
	}

	public void setHomeAddress(Address homeAddress) {
		this.homeAddress = homeAddress;
	}

	public Address getWorkAddress() {
		return workAddress;
	}

	public void setWorkAddress(Address workAddress) {
		this.workAddress = workAddress;
	}

	public String toString()
```

```
 {
                return String.format("name - [%s] home
address - [%s] work address - [%s])"
                                    , name, homeAddress,
workAddress);

        }

}
```

/src/com/in28minutes/oops/level2/CustomerRunner.java

```
package com.in28minutes.oops.level2;

public class CustomerRunner {

        public static void main(String[] args) {
                Address homeAddress = new Address("line 1",
"Hyderabad", "500035");
                Customer customer = new Customer("Ranga",
homeAddress);
                System.out.println(customer);

                Address workAddress = new Address("line 1
for work", "Hyderabad", "500078");
                customer.setWorkAddress(workAddress);

                System.out.println(customer);
        }
}
```

/src/com/in28minutes/oops/level2/Fan.java

```
package com.in28minutes.oops.level2;
```

```
public class Fan {

	//state
	private String make;
	private double  radius;
	private String color;



	private boolean isOn;
	private byte speed; //0 to  5

	//creation
	public Fan(String make, double radius, String
color) {
		this.make = make;
		this.radius = radius;
		this.color = color;
	}

	public void switchOn() {
		this.isOn = true;
		setSpeed((byte)5);
	}

	public void switchOff() {
		this.isOn = false;
		setSpeed((byte)0);
	}

	public void setSpeed(byte speed) {
		this.speed = speed;
	}

	//print the state
	public String toString()
```

```
 {
                return String.format("make - %s, radius -
%f , color - %s , isOn - %b , speed - %d",
                                make, radius, color, isOn,
speed);
        }

}
```

/src/com/in28minutes/oops/level2/FanRunner.java

```
package com.in28minutes.oops.level2;

public class FanRunner {
        public static void main(String[] args) {
                Fan fan = new Fan("Manufacturer 1",
0.34567, "GREEN");
                fan.switchOn();
                System.out.println(fan);
                fan.setSpeed((byte)3);
                System.out.println(fan);
                fan.switchOff();
                System.out.println(fan);
        }
}
```

/src/com/in28minutes/oops/level2/inheritance/AnimalRunner.java

```
package com.in28minutes.oops.level2.inheritance;

abstract class Animal {
        abstract void bark();
}
```

```
class Dog extends Animal {
        public void bark() {
                System.out.println("Bow Bow");
        }
}
 class Cat extends Animal {
        public void bark()

 {
                System.out.println("Meow Meow");
        }
}

public class AnimalRunner {
        public static void main(String[] args) {
                Animal[] animals = {new Cat(), new Dog()};
                for(Animal animal:animals) {
                        animal.bark();
                }

        }

}
```

/src/com/in28minutes/oops/level2/inheritance/Employee.java

```
package com.in28minutes.oops.level2.inheritance;

import java.math.BigDecimal;

public class Employee extends Person {
        private String title;
        private String employerName;
        private char employeeGrade;
        private BigDecimal salary;
```

```
        public Employee(String name, String title) {
                super(name);
                this.title = title;
                System.out.println("Employee Constructor");
        }

        public String getTitle() {
                return

title;
        }

        public void setTitle(String title) {
                this.title = title;
        }


        public String getEmployerName() {
                return employerName;
        }

        public void setEmployerName(String employerName) {
                this.employerName = employerName;
        }

        public char getEmployeeGrade() {
                return employeeGrade;
        }

        public void setEmployeeGrade(char employeeGrade) {
                this.employeeGrade = employeeGrade;
        }

        public BigDecimal getSalary() {
                return salary;
        }
```

```
        public void setSalary(BigDecimal salary) {
                this.salary = salary;
        }

        public String toString()  {
        return super.toString() + "#" + title + "#" +
employerName + "#" + employeeGrade;
    }

}
```

/src/com/in28minutes/oops/level2/inheritance/Person.java

```
package com.in28minutes.oops.level2.inheritance;
 public class Person extends Object{

        private String name;
        private String email;
        private String phoneNumber;

        public Person(String name) {
                System.out.println("Person Constructor");
                this.name = name;
        }

        public String getName() {
                return name;
        }

        public String getEmail() {
                return email;
        }

        public void setEmail(String email) {
                this.email = email;
        }
```

```
        public String getPhoneNumber() {
                return phoneNumber;
        }

        public void setPhoneNumber(String phoneNumber) {
                this.phoneNumber = phoneNumber;

        }


        public String toString()  {
        return name + "#" + email + "#" + phoneNumber;
    }
}
```

/src/com/in28minutes/oops/level2/inheritance/Student.java

```
package com.in28minutes.oops.level2.inheritance;

public class Student extends Person {
        private String collegeName;
        private int year;

        public Student(String name, String collegeName) {
                super(name);
                this.collegeName = collegeName;
        }

        public String getCollegeName() {
                return collegeName;
        }

        public void setCollegeName(String collegeName) {
                this.collegeName = collegeName;
        }
```

```
        public int getYear() {
                return year;
        }

        public void setYear(int year) {
                this.year = year;
        }

}
```

/src/com/in28minutes/oops/level2/inheritance/StudentRunner.java

```
package com.in28minutes.oops.level2.inheritance;

public class StudentRunner {


        public static void main(String[] args) {

                //Student student = new Student();
                //student.setName("Ranga");

//student.setEmail("in28minutes@gmail.com");

                /*
                Person person = new Person();
                person.setName("Ranga");
                person.setEmail("ranga@in28minutes.com");
                person.setPhoneNumber("123-456-7890");
                String value = person.toString();
                System.out.println(value);
                System.out.println(person);
                */

                Employee employee = new Employee("Ranga",
```

```
 "Programmer Analyst");
                //employee.setName("Ranga");
                employee.setEmail("ranga@in28minutes.com");
                employee.setPhoneNumber("123-456-7890");
                employee.setEmployeeGrade('A');
                employee.setTitle("Programmer Analyst");



        System.out.print(employee);


        } }
```

/src/com/in28minutes/oops/level2/inheritance/StudentWithoutInheritance.java

```
package com.in28minutes.oops.level2.inheritance;

public class StudentWithoutInheritance {

        private String name;
        private String email;
        private String phoneNumber;

        private String college;
        private int year;

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }
```

```
        public String getEmail() {
                return email;
        }

        public void setEmail(String email) {
                this.email = email;
        }

        public String getPhoneNumber()

{
                return phoneNumber;

        }

        public void setPhoneNumber(String phoneNumber) {
                this.phoneNumber = phoneNumber;
        }

        public String getCollege()  {
                return college;
        }

        public void setCollege(String college)

{
                this.college = college;
        }

        public int getYear() {
                return year;
        }

        public void setYear(int year) {
                this.year = year;
```

```
        }



}
```

/src/com/in28minutes/oops/level2/interfaces/ChessGame.java

```
package com.in28minutes.oops.level2.interfaces;

public class ChessGame implements GamingConsole{


        @Override
        public void up()

 {
                System.out.println("Move piece up");
        }

        @Override
        public void down() {
                System.out.println("Move piece down");
        }

        @Override
        public void left() {
                System.out.println("Move piece left");
        }


        @Override
        public void right() {
                System.out.println("Move piece right");
        }

}
```

/src/com/in28minutes/oops/level2/interfaces/ComplexAlgorithm.java

```
package com.in28minutes.oops.level2.interfaces;

public interface ComplexAlgorithm {
        int complexAlgorithm(int number1, int number2);
}
```

/src/com/in28minutes/oops/level2/interfaces/DummyAlgorithm.java

```
package com.in28minutes.oops.level2.interfaces;

public class DummyAlgorithm implements ComplexAlgorithm{

        @Override
        public int complexAlgorithm(int number1, int
number2) {
                return number1 + number2;
        }

}
```

/src/com/in28minutes/oops/level2/interfaces/FlyableRunner.java

```
package com.in28minutes.oops.level2.interfaces;

interface Flyable{
        void fly();
}
```

```
class Bird implements Flyable{

	@Override
	public void fly() {
		System.out.println("with wings");
	}

}

class Aeroplane implements Flyable{


	@Override
	public void fly() {
		System.out.println("with fuel");
	}
	 }

public class FlyableRunner {

	public static void main(String[] args) {
		Flyable[] flyingObjects = { new Bird(), new
Aeroplane()};
		for(Flyable object : flyingObjects) {
			object.fly();
		}
	}

}
```

/src/com/in28minutes/oops/level2/interfaces/GameRunner.java

```
package com.in28minutes.oops.level2.interfaces;

public class GameRunner {
```

```
        public static void main(String[] args) {
                GamingConsole[] games = {new MarioGame(),
new  ChessGame()};

                for(GamingConsole game:games) {
                        game.up();
                        game.down();
                        game.left();
                        game.right();
                }


        }

}
```

/src/com/in28minutes/oops/level2/interfaces/GamingConsole.java

```
package com.in28minutes.oops.level2.interfaces;
 public interface GamingConsole {
        public void up();
        public void down();
        public void left();
        public void right();
}
```

/src/com/in28minutes/oops/level2/interfaces/MarioGame.java

```
package com.in28minutes.oops.level2.interfaces;

public class MarioGame implements GamingConsole{

        @Override
        public void up()
```

```
 {
                System.out.println("Jump");
        }

        @Override
        public void down() {
                System.out.println("Goes into a hole");
        }

        @Override
        public void left() {
        }


        @Override
        public void right() {
                System.out.println("Go Forward");

        }

}
```

/src/com/in28minutes/oops/level2/interfaces/Project.java

```
package com.in28minutes.oops.level2.interfaces;

public class Project {

        interface Test {
                void

 nothing();

                default void nothing1() {

                }
```

```
        }

        class Class1 implements Test  {

                @Override
                public void nothing() {
                        // TODO Auto-generated method stub

                }

        }

        class Class2 implements Test

 {

                @Override
                public void nothing() {
                        // TODO Auto-generated method

 stub

                }

        }

        public static void main(String[] args)  {
                ComplexAlgorithm algorithm = new
RealAlgorithm();

System.out.println(algorithm.complexAlgorithm(10, 20));
        }

}
```

/src/com/in28minutes/oops/level2/interfaces/RealAlgorithm.java

```
package com.in28minutes.oops.level2.interfaces;

public class RealAlgorithm implements ComplexAlgorithm{

        @Override
        public int complexAlgorithm(int number1, int
number2) {
                return number1 * number2;
        }

}
```

/src/com/in28minutes/oops/level2/RecipeRunner.java

```
package com.in28minutes.oops.level2;
  public class RecipeRunner

 {

        public static void main(String[] args)  {
                Recipe1 recipe = new Recipe1();
                recipe.execute();

                RecipeWithMicrowave recipeWithMicrowave =
new  RecipeWithMicrowave();
                recipeWithMicrowave.execute();

        }

}
```

/src/com/in28minutes/oops/level2/RecipeWithMicrowave.java

```
package com.in28minutes.oops.level2;
 public class RecipeWithMicrowave extends AbstractRecipe{

        @Override
        void getReady() {
                System.out.println("Get the raw
materials");
                System.out.println("Switch on the
microwave");
        }

        @Override

        void doTheDish() {
                System.out.println("get stuff ready");
                System.out.println("Put it in the
microwave");
        }



        @Override
        void cleanup() {
                System.out.println("Cleanup the
utensils");
                System.out.println("Switch off the
microwave");
        }
 }
```

/src/com/in28minutes/oops/level2/Rectangle.java

```
package com.in28minutes.oops.level2;
```

```
public class Rectangle {

	//state
	private int length;

	private int width;

	//creation
	public Rectangle(int length, int width) {
		this.length = length;
		this.width = width;
	}



	//operations

	public int getLength() {
		return length;
	}

	public void setLength(int length) {
		this.length = length;


	}

	public int getWidth() {
		return width;
	}

	public void setWidth(int width) {
		this.width = width;
	}

	public int area()  {
		return length * width;
```

```
        }

        public int perimeter() {
                return 2 * (length + width);
        }

        public String toString() {
                return String.format("length - %d width -

 %d area - %d perimeter - %d",
                                length, width, area(),
perimeter());
        }
}
```

/src/com/in28minutes/oops/level2/RectangleRunner.java

```
package com.in28minutes.oops.level2;

public class RectangleRunner {

        public static void main(String[] args) {
                Rectangle rectangle = new Rectangle(12,

 23);
                System.out.println(rectangle);
                rectangle.setWidth(25);
                System.out.println(rectangle);
        }
 }
```

/src/com/in28minutes/oops/level2/Review.java

```
package com.in28minutes.oops.level2;
 public class Review {
```

```
        private int id;
        private String description;
        private int rating;

        public Review(int id, String description, int
rating) {
                this.id =

 id;
                this.description = description;
                this.rating = rating;
        }

        public String toString() {
                return id + " " + description + " " +
rating;
        }

}
```

/src/com/in28minutes/oops/level2/Recipe1.java

```
package com.in28minutes.oops.level2;

public class Recipe1 extends AbstractRecipe {

    @Override
    void getReady() {
        System.out.println("Get the raw materials");
        System.out.println("Get the utensils");
    }

    @Override
    void doTheDish() {
        System.out.println("do the dish");
```

```
 }

    @Override
    void cleanup() {
        System.out.println("Cleanup the  utensils");
    }

}
```

Collections

Steps

- Step 01 - Java Collections - Section Overview with Need For Collections
- Step 02 - List Interface - Introduction - Position is King
- Step 03 - List Inteface - Immutability and Introduction of Implementations - ArrayList, LinkedList and Vector
- Step 04 - List Inteface Implementations - ArrayList vs LinkedList
- Step 05 - List Inteface Implementations - ArrayList vs Vector
- Step 06 - List Inteface - Methods to add, remove and change elements and lists
- Step 07 - List and ArrayList - Iterating around elements
- Step 08 - List and ArrayList - Choosing iteration approach for printing and deleting elements
- Step 09 - List and ArrayList - Puzzles - Type Safety and Removing Integers
- Step 10 - List and ArrayList - Sorting - Introduction to Collections sort static method
- Step 11 - List and ArrayList - Sorting - Implementing Comparable Inteface in Student Class
- Step 12 - List and ArrayList - Sorting - Providing Flexibility by implementing Comparator interface
- Step 13 - List and ArrayList - A Summary
- Step 14 - Set Interface - Introduction - No Duplication
- Step 15 - Understanding Data Structures - Array, LinkedList and Hashing
- Step 16 - Understanding Data Structures - Tree - Sorted Order
- Step 17 - Set Interface - Hands on - HashSet, LinkedHashSet and TreeSet
- Step 18 - Set Interface - Exercise - Find Unique Characters in a List
- Step 19 - TreeSet - Methods from NavigableSet - floor,lower,upper, subSet, head and tailSet
- Step 20 - Queue Interface - Process Elements in Order

- Step 21 - Introduction to PriorityQueue - Basic Methods and Customized Priority
- Step 22 - Map Interface - An Introduction - Key and Value
- Step 23 - Map Interface - Implementations - HashMap, HashTable, LinkedHashMap and TreeMap
- Step 24 - Map Interface - Basic Operations
- Step 25 - Map Interface - Comparison - HashMap vs LinkedHashMap vs TreeMap
- Step 26 - Map Interface - Exercise - Count occurances of characters and words in a piece of text
- Step 27 - TreeMap - Methods from NavigableMap - floorKey, higherKey, firstEntry, subMap and more
- Step 28 - Java Collections - Conclusion with Three Tips

Exercises

Set Interface

- Find unique characters in a list of characters
  - In Insertion Order
  - In Sort Order

Map Interface

- Find number of occurances of each character and word in a piece of text.

Code Examples

/12-Collections/commands.txt

```
words.size()
words.isEmpty()
words.get(0)
words.contains("Dog");
words.contains("Cat");
words.indexOf("Cat")
words
words.indexOf("Dog")
words.add("Dog")
```

```
List<String> wordsLinkedList = new LinkedList<String>
(words);
List<String> wordsVector = new Vector<String>(words);
wordsArrayList.add("Dog")
wordsArrayList
wordsArrayList.add("Elephant")
wordsArrayList.add(2, "Ball")
wordsArrayList
wordsArrayList.add("Ball")
wordsArrayList
List<String> newList = List.of("Yak","Zebra");
wordsArrayList.addAll(newList)
wordsArrayList
wordsArrayList.set(6, "Fish")
wordsArrayList
wordsArrayList.remove(2)
wordsArrayList
wordsArrayList.remove("Dog")
wordsArrayList
wordsArrayList.remove("Dog")
for(int i=0; i < words.size(); i++) {
     System.out.println(words.get(i));
}
for(String word:words) {
     System.out.println(word);
}
Iterator wordsIterator = words.iterator();
while(wordsIterator.hasNext()) {
      System.out.println(wordsIterator.next());
}
List<String> wordsArrayList = new ArrayList<String>(words);
for(String word:words) {
   if(word.endsWith("at")) {
        words.remove(word);
   }
}
```

```
for(String word:wordsArrayList) {
   if(word.endsWith("at")) {
        wordsArrayList.remove(word);
   }
}
wordsArrayList
for(String word:words) {
   if(word.endsWith("at"))
      System.out.println(word);
}
for(String word:wordsAl) {
   if(word.endsWith("at")) {
        words.remove(word);
   }
}
for(String word:wordsAl) {
   if(word.endsWith("at")) {
        wordsAl.remove(word);
   }
}
wordsAl
List<String> words = List.of("Apple", "Bat" , "Cat");
List<String> wordsAl = new ArrayList<>(words);
Iterator<String> iterator = wordsAl.iterator();
while(iterator.hasNext()) {
     if(iterator.next().endsWith("at")) {
          iterator.remove();
     }
}
wordsAl
List value = List.of("A", 'A' , 1, 1.0);
value.get(2)
value.get(2) instanceof Integer
value.get(1) instanceof Character
value.get(3) instanceof Double
numbers.indexOf(101);
```

```
numbersAl.indexOf(101)
numbersAl.remove(101);
numbersAl.remove(101)
numbersAl.remove(Integer.valueOf(101))
numbersAl
List<Integer> numbersAl = new ArrayList<>(numbers);
Collections.sort(numbersAl);
numbersAl
set.add("Apple");
Set<String> hashset = new HashSet<>(set);
hashset.add("Apple")
hashset
Set<String> set = Set.of("A", "Z","D", "C", "B");
hashSet.add("A")
hashSet.add("B")
hashSet
hashSet.add("C")
hashSet
hashSet.add("CA")
hashSet
hashSet.add("CAfsadfa")
hashSet
treeSet.add("Cat")
treeSet.add("Bat")
treeSet.add("Apple")
treeSet
hashSet.add("Cat");
hashSet.add("Bat");
hashSet.add("Apple");
hashSet
hashSet.add("Dog");
hashSet
hashSet.add("Elephant");
hashSet.add(5);
hashSet.add(4);
hashSet.add(3);
```

```
hashSet.add(2);
hashSet.add(1);
hashSet
Set<Integer> hashSet = new HashSet<>();
hashSet.add(589789);
hashSet.add(58978);
hashSet.add(5897);
hashSet.add(589);
hashSet.add(58);
hashSet.add(5);
hashSet
Set<Integer> linkedHashSet = new LinkedHashSet<>();
linkedHashSet.add(589789);
linkedHashSet.add(58978);
linkedHashSet.add(5897);
linkedHashSet.add(589);
linkedHashSet.add(58);
linkedHashSet.add(5);
linkedHashSet
Set<Integer> treeSet = new TreeSet<>();
treeSet.add(584567);
treeSet.add(58456);
treeSet.add(5845);
treeSet.add(584);
treeSet.add(58);
treeSet.add(5);
treeSet
numbers.add(765432);
numbers.add(76543);
numbers.add(7654);
numbers.add(765);
numbers.add(76);
numbers
numbers.add(765432);
numbers.add(76543);
numbers.add(7654);
```

```
numbers.add(765);
numbers.add(76);
numbers
numbers.add(765789);
numbers
numbers.add(76)
numbers
numbers.add(765432);
numbers.add(76543);
numbers.add(7654);
numbers.add(765);
numbers.add(76);
numbers
numbers.add(76)
List<Character> characters = List.of('A','Z','A', 'B',
'Z','F');
TreeSet<Integer> numbers = new TreeSet<>
(Set.of(65,54,34,12,99));
numbers.floor(40)
numbers.floor(34)
numbers.lower(34)
numbers.ceiling(34)
numbers.ceiling(36)
numbers.higher(34)
numbers
numbers.subSet(20,80)
numbers.subSet(34,54)
numbers.subSet(34,65)
numbers.subSet(34,true,65,true)
numbers.subSet(34,false,65,false)
numbers.headSet(50)
numbers.tailSet(50)
Queue<String> queue = new PriorityQueue<>();
queue.poll()
queue.offer("Apple")
queue.addAll(List.of("Zebra", "Monkey", "Cat"))
```

```
queue
queue.poll()
queue
queue.poll()
queue.poll()
queue.poll()
queue.poll()
map.put("R",1);
map
map.get("Z")
map.get("A")
map.get("C")
map.size()
map.isEmpty()
map.containsKey("A")
map.containsKey("F")
map.containsValue(3)
map
map.containsValue(4)
map.keySet()
map.values()
map
hashmap.put("F",5)
hashmap
hashmap.put("Z",11)
hashmap
Map<String, Integer> map = Map.of("A",3,"Z",5,"B",10);
Map<String, Integer> linkedHashmap = new LinkedHashMap<>
(map);
HashMap<String, Integer> hashmap = new HashMap<>();
hashmap.put("Z",5)
hashmap.put("A",15)
hashmap.put("F",25)
hashmap.put("L",250)
hashmap
LinkedHashMap<String, Integer> linkedHashMap = new
```

```
 LinkedHashMap<>();
hashmap
linkedHashMap.put("F",25)
linkedHashMap.put("A",15)
linkedHashMap.put("Z",5)
linkedHashMap.put("L",250)
linkedHashMap
treemap.put("F",25)
treemap.put("A",15)
treemap.put("Z",5)
treemap.put("L",250)
treemap
TreeMap<String, Integer> treemap = new TreeMap<>();
treemap.put("F",25)
treemap.put("Z",5)
treemap.put("L",250)
treemap.put("A",15)
treemap.put("B",25)
treemap.put("G",25)
treemap
treemap.higherKey("C")
treemap.lowerKey("C")
treemap.lowerKey("B")
treemap.floorKey("B")
treemap.higherKey("B")
treemap.higherKey("C")
treemap.ceilingKey("B")
treemap.lowerKey("B")
treemap.floorKey("B")
treemap.firstEntry()
treemap.lastEntry()
treemap
treemap.subMap("C", "Y")
treemap.subMap("B", "Z")
treemap.subMap("B",true, "Z",true)
```

/12-Collections/src/collections/MapRunner.java

```
package collections;

import java.util.HashMap;
import java.util.Map;


public class MapRunner {

	public static void main(String[] args) {
		String str = "This is an awesome occasion.
"
				+ "This has never happened
before.";

		Map<Character, Integer> occurances = new
HashMap<>();

		char[] characters = str.toCharArray();

		for(char character:characters) {
			//Get the character
			Integer integer =
occurances.get(character);
			if(integer==null) {
				occurances.put(character,
1);
			} else {
				occurances.put(character,
integer + 1);
			}
		}
```

```
        System.out.println(occurances);

                Map<String, Integer> stringOccurances = new
HashMap<>();
                String[] words = str.split(" ");


                for(String word:words) {
                        //Get the character
                        Integer integer =
stringOccurances.get(word);
                        if(integer==null) {
                                stringOccurances.put(word,
1);
                        } else {
                                stringOccurances.put(word,
integer + 1);
                        }
                }

                System.out.println(stringOccurances);


        }

}
```

/12-Collections/src/collections/QueueRunner.java

```
package collections;

import java.util.Comparator;
import java.util.List;
import java.util.PriorityQueue;
import java.util.Queue;
```

```
class StringLengthComparator implements Comparator<String>
{
        @Override
        public int compare(String value1, String value2) {
                return Integer.compare(value2.length(),
                                value1.length());
        }

}

public class QueueRunner {

        public static void main(String[] args) {
                Queue<String> queue = new PriorityQueue<>
(10,
                                new
StringLengthComparator());
                queue.addAll(List.of("Zebra", "Monkey",
"Cat", "A",
                                "B", "C", "D", "E", "F",
"G"));
                queue.offer("Z");
                while (queue.peek() != null)
                        System.out.println(queue.poll());
        }

}
```

/12-Collections/src/collections/SetRunner.java

```
package collections;

import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;
```

```
import java.util.TreeSet;

public class SetRunner {

        public static void main(String[] args) {
                List<Character> characters =
List.of('A','Z','A', 'B', 'Z','F');
                //unique - Set
                // Tree
                // Hash
                // LinkedHash

                Set<Character> treeSet = new TreeSet<>
(characters);
                System.out.println("treeSet "+ treeSet);

                Set<Character> linkedHashSet = new
LinkedHashSet<>(characters);
                System.out.println("linkedHashSet "+
linkedHashSet);

                Set<Character> hashSet = new HashSet<>
(characters);
                System.out.println("hashSet "+ hashSet);



        }

}
```

/12-Collections/src/collections/Student.java

```
package collections;

public class Student implements Comparable<Student>{
```

```
        private int id;
        private String name;

        public Student(int id, String name) {
                super();
                this.id = id;
                this.name = name;
        }

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String toString() {
                return id + " " + name;
        }

        @Override
        public int compareTo(Student that) {
                return Integer.compare(that.id, this.id);
        }
}
```

/12-Collections/src/collections/StudentsCollectionRunner.java

```
package collections;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

class AscendingStudentComparator implements
Comparator<Student> {
	@Override
	public int compare(Student student1, Student
student2) {
		return Integer.compare(student1.getId(),
student2.getId());
	}
}

public class StudentsCollectionRunner {

	public static void main(String[] args) {
		List<Student> students = List.of(new
Student(1, "Ranga"),
				new Student(100, "Adam"),
				new Student(2,"Eve"));

		ArrayList<Student> studentsAl = new
ArrayList<>(students);

		System.out.println(studentsAl);

		Collections.sort(studentsAl);
		System.out.println("Desc " + studentsAl);

		//Collections.sort(studentsAl, new
```

```
 AscendingStudentComparator());

                studentsAl.sort(new
AscendingStudentComparator());

System.out.println("AscendingStudentComparator " +
studentsAl);
        }

}
```

Generics

Steps

- Step 01 - Introduction to Generics - Why do we need Generics?
- Step 02 - Implementing Generics for the Custom List
- Step 03 - Extending Custom List with a Generic Return Method
- Step 04 - Generics Puzzles - Restrictions with extends and Generic Methods
- Step 05 - Generics and WildCards - Upper Bound and Lower Bound

Code Examples

/13-Generics/src/com/in28minutes/generics/GenericsRunner.java

```
package com.in28minutes.generics;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class GenericsRunner {

	static <X> X doubleValue(X value) {
		return value;
	}

	static <X extends List> void duplicate(X list) {
		list.addAll(list);
	}
```

```
        static double sumOfNumberList(
                        List<? extends Number> numbers) {
                double sum = 0.0;
                for (Number number : numbers) {
                        sum += number.doubleValue();
                }
                return sum;
        }

        static void addACoupleOfValues(
                        List<? super Number> numbers) {
                numbers.add(1);
                numbers.add(1.0);
                numbers.add(1.0f);
                numbers.add(1l);
        }

        public static void main(String[] args) {
                List emptyList = new ArrayList<Number>();
                addACoupleOfValues(emptyList);
                System.out.println(emptyList);

                System.out.println(
                                sumOfNumberList(List.of(1,
2, 3, 4, 5)));
                System.out.println(sumOfNumberList(
                                List.of(1.1, 2.1, 3.1, 4.1,
5.1)));
                System.out.println(sumOfNumberList(
                                List.of(1l, 2l, 3l, 4l,
5l)));

                String value1 = doubleValue(new String());
                Integer number1 =
doubleValue(Integer.valueOf(5));
                ArrayList list1 = doubleValue(new
```

```
 ArrayList());

                LinkedList<Integer> numbers = new
LinkedList<>(
                                    List.of(1, 2, 3));
                duplicate(numbers);
                System.out.println(numbers);

                MyCustomList<String> list = new
MyCustomList<>();
                list.addElement("Element 1");
                list.addElement("Element 2");
                String value = list.get(0);

                System.out.println(value);

                MyCustomList<Integer> list2 = new
MyCustomList<>();
                list2.addElement(Integer.valueOf(5));
                list2.addElement(Integer.valueOf(7));
                Integer number = list2.get(0);
                System.out.println(number);

        }

}
```

/13-Generics/src/com/in28minutes/generics/MyCustomList.java

```
package com.in28minutes.generics;

import java.util.ArrayList;

public class MyCustomList<T>{

```

```
        ArrayList<T> list = new ArrayList<>();

        public void addElement(T element) {
                list.add(element);
        }

        public void removeElement(T element) {
                list.remove(element);
        }

        public String toString() {
                return list.toString();
        }

        public T get(int index) {
                return list.get(index);
        }
}
```

FunctionalProgramming

Steps

- Step 01 - Introduction to Functional Programming - Functions are First Class Citizens
- Step 02 - Functional Programming - First Example with Function as Parameter
- Step 03 - Functional Programming - Exercise - Loop a List of Numbers
- Step 04 - Functional Programming - Filtering - Exercises to print odd and even numbers from List
- Step 05 - Functional Programming - Collect - Sum of Numbers in a List
- Step 06 - Functional Programming vs Structural Programming - A Quick Comparison
- Step 07 - Functional Programming Terminology - Lambda Expression, Stream and Operations on a Stream
- Step 08 - Stream Intermediate Operations - Sort, Distinct, Filter and Map
- Step 09 - Stream Intermediate Operations - Exercises - Squares of First 10, Map String List to LowerCase and Length of String
- Step 10 - Stream Terminal Operations - 1 - max operation with Comparator
- Step 11 - Stream Terminal Operations - 2 - min, collect to List,
- Step 12 - Optional class in Java - An Introduction
- Step 13 - Behind the Screens with Functional Interfaces - Implement Predicate Interface
- Step 14 - Behind the Screens with Functional Interfaces - Implement Consumer Interface
- Step 15 - Behind the Screens with Functional Interfaces - Implement Function Inteface for Mapping
- Step 16 - Simplify Functional Programming code with Method References - static and instance methods
- Step 17 - Functions are First Class Citizens

- Step 18 - Introduction to Functional Programming - Conclusion

Exercises

- Create a list of numbers
  - Print each element using forEach
  - Filter Odd and Even numbers
- Print squares of first 10 Numbers using `IntStream.range(1,11)`
- `List.of("Apple","Ant","Bat").stream()`
  - Map each to lowercase and print
  - Print length (no of characters) of each word using map

Code Examples

/14-FunctionalProgramming/commands.txt

```
List<Integer> list = List.of(1,4,7,9);
list.stream().forEach(
      element -> System.out.println(element)
)
list.stream().filter(
               element -> element%2 == 1)
list.stream().filter(
               element -> element%2 == 1).
             forEach(
               element -> System.out.println(element))
list.stream().filter(element ->
element%2==1).forEach(element->System.out.println(element))
list.stream().filter(element ->
element%2==0).forEach(element->System.out.println(element))
numbers.stream().sorted().forEach(e-
>System.out.println(e));
List<Integer> numbers = List.of(3,5,3,213,45,5,7);
numbers.stream().distinct().forEach(e-
>System.out.println(e));
```

```
numbers.stream().distinct().sorted().forEach(e-
>System.out.println(e));
numbers.stream().distinct().map(e -> e * e).forEach(e-
>System.out.println(e));
IntStream.range(1,10).forEach(p->System.out.println(p))
IntStream.range(1,11).forEach(p->System.out.println(p))
IntStream.range(1,11).map(e -> e * e).forEach(p-
>System.out.println(p)) List.of("Apple", "Ant",
"Bat").stream().map(s->s.toLowerCase()).forEach(p ->
System.out.println(p))
List.of("Apple", "Ant", "Bat").stream().map(s-
>s.length()).forEach(p -> System.out.println(p))
IntStream.range(1,11).reduce( 0 , (n1,n2) -> n1+n2)
List.of(23,12,34,53).stream().max((n1,n2) ->
Integer.compare(n1,n2))
$24.isPresent()
List.of(23,12,34,53).stream().max((n1,n2) ->
Integer.compare(n1,n2)).get()
List.of(23,12,34,53).stream().max((n1,n2) ->
Integer.compare(n1,n2)).get()
List.of(23,12,34,53).stream().max((n1,n2) ->
Integer.compare(n1,n2)).get()
List.of(23,12,34,53).stream().min((n1,n2) ->
Integer.compare(n1,n2)).get()
List.of(23,12,34,53).stream().filter(e -> e%2==1).forEach(e
-> System.out.println(e))
List.of(23,12,34,53).stream().filter(e ->
e%2==1).collect(Collectors.toList())
List.of(23,12,34,53).stream().filter(e ->
e%2==0).collect(Collectors.toList())
IntStream.range(1,11).map(e -> e * e)
List.of(23,12,34,53).stream().filter(e ->
e%2==0).collect(Collectors.toList())
IntStream.range(1,10).map(e -> e * e)
IntStream.range(1,10).map(e -> e *
e).boxed().collect(Collectors.toList())
```

```
IntStream.range(1,11).map(e -> e *
e).boxed().collect(Collectors.toList())
List.of(23,45,67,12).stream().filter(n -> n%2==0).max(
(n1,n2) -> Integer.compare(n1,n2) )
$39.get()
$39.isPresent()
List.of(23,45,67).stream().filter(n -> n%2==0).max( (n1,n2)
-> Integer.compare(n1,n2) )
$42.isPresent()
$42.orElse(0)
$42.orElse(0)
List.of(23,45,67).stream().filter(n -> n%2==0).max( (n1,n2)
-> Integer.compare(n1,n2) ).orElse(0)
List.of(23,45,67,34).stream().filter(n -> n%2==0).max(
(n1,n2) -> Integer.compare(n1,n2) ).orElse(0)
```

/14-FunctionalProgramming/src/com/in28minutes/functionalprogramming/FPNumberRunner.java

```
package com.in28minutes.functionalprogramming;

import java.util.List;

public class FPNumberRunner {

        public static void main(String[] args) {
                List<Integer> numbers =
List.of(4,6,8,13,3,15);

                //Exercise - Print squares of first 10
numbers!
                //Clue - IntStream.range(1,10)

                //List.of("Apple", "Ant", "Bat").stream()
                //Map all of these to lowercase and print
```

```
 them

                //List.of("Apple", "Ant", "Bat").stream()
                //Print the length of each element

                /*
                numbers.stream()
                                .forEach( element -
>System.out.println(element));*/

                int sum = fpSum(numbers);

                System.out.println(sum);

        }

        private static int fpSum(List<Integer> numbers) {
                return numbers.stream()
                                .reduce(0,
                                        (number1, number2)
->  number1 + number2);
        }

        private static int normalSum(List<Integer> numbers)
{
                int sum = 0;
                for(int number:numbers) {
                        sum += number;
                }
                return sum;
        }

}
```

/14-FunctionalProgramming/src/com/in28minutes/functionalprogramming/FunctionalProgrammingRunner.java

```
package com.in28minutes.functionalprogramming;

import java.util.List;
 public class FunctionalProgrammingRunner {

	public static void main(String[] args) {
		List<String> list = List.of("Apple", "Bat",
"Cat",
				"Dog");
		printWithFPWithFiltering(list);

	}

	private static void printBasic(List<String> list) {
		for (String string : list) {
			System.out.println(string);
		}
	}

	private static void printBasicWithFiltering(
			List<String> list) {
		for (String string : list) {
			if (string.endsWith("at")) {
				System.out.println(string);
			}
		}
	}

	private static void printWithFP(List<String> list)
{
		list.stream().forEach(element ->
```

```
 System.out
                                    .println("element - " +
element));
        }

        private static void
printWithFPWithFiltering(List<String> list)  {
                list.stream()
                                    .filter(element ->
element.endsWith("at"))
                        .forEach(element ->
System.out.println("element - " + element));
        }

}
```

/14-FunctionalProgramming/src/com/in28minutes/functionalprogramming/LambdaBehindTheScreensRunner.java

```
package com.in28minutes.functionalprogramming;

import java.util.List;
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;

class EvenNumberPredicate implements Predicate<Integer> {

        @Override
        public boolean test(Integer number) {
                return number%2 == 0;
        }

}
```

```
class SystemOutConsumer implements Consumer<Integer> {

	@Override
	public void accept(Integer number) {
		System.out.println(number);
	}

}

class NumberSquareMapper implements Function<Integer,
Integer> {

	@Override
	public Integer apply(Integer number) {
		return number * number;
	}

}

public class LambdaBehindTheScreensRunner {



	public static void main(String[] args) {

		List.of(23,43,34,45,36,48).stream()
		.filter(n -> n%2 ==0)
		.map(n -> n * n)
		.forEach(e -> System.out.println(e));

		List.of(23,43,34,45,36,48).stream()
		.filter(new EvenNumberPredicate())
		.map(new NumberSquareMapper())
		.forEach(new SystemOutConsumer());

		//.map(n -> n *
```

```
 n)
                //<R> Stream<R> map(Function<? super T, ?
extends R> mapper);
                // R apply(T  t);

                //.filter(new EvenNumberPredicate())
                //Stream<T> filter(Predicate<? super T>
predicate);
                //boolean test(T t);

                //.forEach(e -> System.out.println(e));
                //Consumer<? super T> action
                //void accept(T t);

                //1.Storing functions in variables
                //2.Passing functions to methods <=
                //3.Returning functions from methods

                Predicate<? super Integer> evenPredicate =
createEvenPredicate();
                Predicate<? super Integer> oddPredicate = n
-> n%2 ==1;

                List.of(23,43,34,45,36,48).stream()
                                .filter(evenPredicate)
                                .map(n -> n * n)
                                .forEach(e ->
System.out.println(e));
        }

        private static Predicate<? super Integer>
createEvenPredicate() {
                return n -> n%2 ==0;
        }

}
```

/14-FunctionalProgramming/src/com/in28minutes/functionalprogramming/MethodReferencesRunner.java

```
package com.in28minutes.functionalprogramming;

import java.util.List;

public class MethodReferencesRunner {

	public static void print(Integer number) {
		System.out.println(number);
	}

	public static void main(String[] args) {
		List.of("Ant", "Bat", "Cat", "Dog",
"Elephant")
				.stream().map(s ->
s.length())
				.forEach(s ->
MethodReferencesRunner
						.print(s));

		List.of("Ant", "Bat", "Cat", "Dog",
"Elephant")

.stream().map(String::length)

.forEach(MethodReferencesRunner::print);

		List.of(23,45,67,34).stream()
				.filter(n -> n%2==0)
				.max( (n1,n2) ->
Integer.compare(n1,n2) )
```

```
        .orElse(0);

                Integer max = List.of(23, 45, 67,
34).stream()

.filter(MethodReferencesRunner::isEven)
                                .max(Integer::compare)
                                .orElse(0);

                System.out.println(max);
        }

        public static boolean isEven(Integer number) {
                return number % 2 == 0;
        }

}
```

Threads And Concurrency

Steps

- Step 01 - Introduction to Threads and MultiThreading - Need for Threads
- Step 02 - Creating a Thread for Task1 - Extending Thread Class
- Step 03 - Creating a Thread for Task2 - Implement Runnable Interface
- Step 04 - Theory - States of a Thread
- Step 05 - Placing Priority Requests for Threads
- Step 06 - Communication between Threads - join method
- Step 07 - Thread utility methods and synchronized keyword - sleep, yield
- Step 08 - Need for Controlling the Execution of Threads
- Step 09 - Introduction to Executor Service
- Step 10 - Executor Service - Customizing number of Threads
- Step 11 - Executor Service - Returning a Future from Thread using Callable
- Step 12 - Executor Service - Waiting for completion of multiple tasks using invokeAll
- Step 13 - Executor Service - Wait for only the fastest task using invokeAny
- Step 14 - Threads and MultiThreading - Conclusion

Code Examples

/15-ThreadsAndParallelism/src/ExecutorServiceRunner.java

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Task extends Thread {


        private int number;
```

```
        public Task(int number) {
                this.number = number;
        }

        public void run() {  //SIGNATURE
                System.out.print("\nTask " + number + "
Started");

                for(int i=number*100;i<=number*100 + 99;
i++)
                        System.out.print(i + " ");

                System.out.print("\nTask" + number +
"Done");
        }
}


public class ExecutorServiceRunner {

        public static void main(String[] args) {
                //ExecutorService executorService =
Executors.newSingleThreadExecutor();
                ExecutorService executorService =
Executors.newFixedThreadPool(5);

                executorService.execute(new Task(1));
                executorService.execute(new Task(2));
                executorService.execute(new Task(3));
                executorService.execute(new Task(4));
                executorService.execute(new Task(5));
                executorService.execute(new Task(6));
                executorService.execute(new
```

```
 Task(7));

                executorService.shutdown();

        }
 }
```

/15-ThreadsAndParallelism/src/MultipleAnyCallableRunner.java

```
import java.util.List;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

public class MultipleAnyCallableRunner {

        public static void main(String[] args) throws
InterruptedException, ExecutionException {
                ExecutorService executorService =
Executors.newFixedThreadPool(3);

                List<CallableTask> tasks = List.of(
                                new
CallableTask("in28Minutes"),
                                new CallableTask("Ranga"),
                                new CallableTask("Adam"));

                String result =
executorService.invokeAny(tasks);

                System.out.println(result);

                executorService.shutdown();
```

```
        }

}
```

/15-ThreadsAndParallelism/src/MultipleCallableRunner.java

```
import java.util.List; import
java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

public class MultipleCallableRunner {

        public static void main(String[] args) throws
InterruptedException, ExecutionException {
                ExecutorService executorService =
Executors.newFixedThreadPool(3);

                List<CallableTask> tasks = List.of(new
CallableTask("in28Minutes"),
                                new CallableTask("Ranga"),
new CallableTask("Adam"));

                List<Future<String>> results =
executorService.invokeAll(tasks);

                for(Future<String> result:results) {
                        System.out.println(result.get());
                }

                executorService.shutdown();
        }

}
```

/15-ThreadsAndParallelism/src/ThreadBasicsRunner.java

```
//extends Thread //implements Runnable

class Task1 extends Thread

 {

        public void run() { //SIGNATURE
                System.out.print("\nTask1 Started");

                for(int i=101;i<=199; i++)
                        System.out.print(i + " ");

                System.out.print("\nTask1 Done");
        }
}

class Task2 implements Runnable {

        @Override
        public void run() {
                System.out.print("\nTask2 Started");

                for(int i=201;i<=299; i++)
                        System.out.print(i + " ");

                System.out.print("\nTask2 Done");


        }

}
```

```
public class ThreadBasicsRunner {



        public static void main(String[] args) throws
InterruptedException

 {

                //•  NEW;
                //• RUNNABLE;
                //•  RUNNING;
                //• BLOCKED/WAITING;
                //• TERMINATED/DEAD;

                //Task1 - 101 to 199
                System.out.print("\nTask1 Kicked Off");
                Task1 task1 = new Task1();
                task1.setPriority(1);
                task1.start(); //task1.run();

                System.out.print("\nTask2 Kicked Off");

                Task2 task2 = new Task2();
                Thread task2Thread = new Thread(task2);
                task2Thread.setPriority(10);
                task2Thread.start();

                task1.join();
                task2Thread.join();

                System.out.print("\nTask3 Kicked Off");

                //Task3
                for(int i=301;i<=399; i++)
                        System.out.print(i + "
```

```
 ");

                System.out.print("\nTask3 Done");

                System.out.print("\nMain Done");

        }

}
```

Exception Handling

Steps

- Step 01 - Introduction to Exception Handling - Your Thought Process during Exception Handling
- Step 02 - Basics of Exceptions - NullPointerException and StackTrace
- Step 03 - Basics of Handling Exceptions - try and catch
- Step 04 - Basics of Handling Exceptions - Exception Hierarchy, Matching and Catching Multiple Exceptions
- Step 05 - Basics of Handling Exceptions - Need for finally
- Step 06 - Basics of Handling Exceptions - Puzzles
- Step 07 - Checked Exceptions vs Unchecked Exceptions - An Example
- Step 08 - Hierarchy of Errors and Exceptions - Checked and Runtime
- Step 09 - Throwing an Exception - Currencies Do Not Match Runtime Exception
- Step 10 - Throwing a Checked Exception - Throws in method signature and handling
- Step 11 - Throwing a Custom Exception - CurrenciesDoNotMatchException
- Step 12 - Write less code with Try with Resources - New Feature in Java 7
- Step 13 - Basics of Handling Exceptions - Puzzles 2
- Step 14 - Exception Handling - Conclusion with Best Practices

Code Examples

/16-ExceptionHandling/src/com/in28minutes/exceptionhandling/CheckedExceptionRunner.java

```
package com.in28minutes.exceptionhandling;
```

```
public class CheckedExceptionRunner {

	public static void main(String[] args) {
	/*	try {
				someOtherMethod();
				Thread.sleep(2000);
			} catch (InterruptedException e)

 {
				e.printStackTrace();
			}*/
			//someOtherMethod1();
			someOtherMethod2();

	}

	private static void someOtherMethod2() throws
RuntimeException{

	}

	private static void someOtherMethod() throws
InterruptedException{
			Thread.sleep(2000);
	}

}
```

/16-ExceptionHandling/src/com/in28minutes/exceptionhandling/ExceptionHandlingRunner.java

```
package com.in28minutes.exceptionhandling;

public class ExceptionHandlingRunner {
```

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

 {
                method1();
                System.out.println("Main Ended");
        }

        private static void method1()

 {
                method2();
                System.out.println("Method1 Ended");
        }

        private static void method2() {
                String str = null;
                str.length();
                System.out.println("Method2 Ended");
        }
}
```

/16-ExceptionHandling/src/com/in28minutes/exceptionhandling/ExceptionHandlingRunner2.java

```
package com.in28minutes.exceptionhandling;

public class ExceptionHandlingRunner2 {

        public static void main(String[] args) {
                method1();
                System.out.println("Main Ended");
        }

        private static void method1() {
                method2();
                System.out.println("Method1 Ended");
```

```
        }

        private static void method2()



 {
                try

 {
                        //String str =  null;
                        //str.length();
                        int[] i =  {1,2};
                        int number = i[3];
                        System.out.println("Method2
Ended");
                } catch (NullPointerException ex) {
                        System.out.println("Matched
NullPointerException");
                        ex.printStackTrace();
                } catch (ArrayIndexOutOfBoundsException ex)
{
                        System.out.println("Matched
ArrayIndexOutOfBoundsException");
                        ex.printStackTrace();
                } catch (Exception ex) {
                        System.out.println("Matched
Exception");
                        ex.printStackTrace();
                }
        }
}
```

/16-ExceptionHandling/src/com/in28minutes/exceptionhandling/FinallyRunner.java

```
package com.in28minutes.exceptionhandling;

import java.util.Scanner;

public class FinallyRunner {



        public static void main(String[] args)

 {

                Scanner scanner =  null;

                try {
                        scanner = new Scanner(System.in);
                        int[] numbers = { 12, 3, 4, 5 };

                        int number = numbers[21];

                } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        System.out.println("Before Scanner
Close");
                        if(scanner!=null) {
                                scanner.close();
                        }
                }

                System.out.println("Just before closing out
main");
        }

}
```

/16-ExceptionHandling/src/com/in28minutes/exceptionhandling/ThrowingExceptionRunner.java

```
package com.in28minutes.exceptionhandling;

class Amount {

	private String currency;


	private int   amount;


	public Amount(String currency, int amount)  {
		super();
		this.currency = currency;
		this.amount = amount;
	}

	public void add(Amount that) throws
CurrenciesDoNotMatchException {

		if(!this.currency.equals(that.currency)) {
			//throw new Exception("Currencies
Don't Match " + this.currency + " & " +that.currency );
			throw new
CurrenciesDoNotMatchException("Currencies Don't Match " +
this.currency + " & " +that.currency );
		}

		this.amount = this.amount + that.amount;
	}

	public String toString() {
		return amount + " " + currency;
```

```
        }
}

class CurrenciesDoNotMatchException extends Exception {
        public CurrenciesDoNotMatchException(String msg) {
                super(msg);
        }
}

public class ThrowingExceptionRunner {



        public static void main(String[] args) throws
CurrenciesDoNotMatchException {
                Amount amount1 = new Amount("USD",

 10);
                Amount amount2 = new Amount("EUR", 20);
                amount1.add(amount2);
                System.out.println(amount1);
        }

}
```

/16-ExceptionHandling/src/com/in28minutes/exceptionhandling/TryWithResourcesRunner.java

```
package com.in28minutes.exceptionhandling;

import java.util.Scanner;

public class TryWithResourcesRunner {

        public static void main(String[] args) {
                try (Scanner scanner = new
```

```
 Scanner(System.in)) {
                            int[] numbers = { 12, 3, 4, 5 };
                            int number = numbers[21];
                   }
          }

}
```

Files

Steps

- Step 01 - List files and folders in Directory with Files list method
- Step 02 - Recursively List and Filter all files and folders in Directory with Step Files walk method and Search with find method
- Step 03 - Read content from a File - Files readAllLines and lines methods
- Step 04 - Writing Content to a File - Files write method
- Step 05 - Files - Conclusion

Code Examples

/17-Files/src/files/DirectoryScanRunner.java

```
package files;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.attribute.BasicFileAttributes;
import java.util.function.BiPredicate;
import java.util.function.Predicate;

public class DirectoryScanRunner {

        public static void main(String[] args) throws
IOException {

                Path currentDirectory =
```

```
 Paths.get(".");

//Files.list(currentDirectory).forEach(System.out::println)
;

                Predicate<? super Path>  predicate
                                = path ->
String.valueOf(path).contains(".java");

                //Files.walk(currentDirectory,
4).filter(predicate).forEach(System.out::println);

                BiPredicate<Path, BasicFileAttributes>
javaMatcher
                = (path,attributes) ->
String.valueOf(path).contains(".java");

                BiPredicate<Path, BasicFileAttributes>
directoryMatcher
                = (path,attributes) ->
attributes.isDirectory();


                Files.find(currentDirectory, 4,
directoryMatcher )

.forEach(System.out::println);

        }

}
```

/17-Files/src/files/FileReadRunner.java

```
package files;


import java.io.IOException;

import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
  public class FileReadRunner {

        public static void main(String[] args) throws
IOException {
                Path pathFileToRead =
Paths.get("./resources/data.txt");

                //List<String> lines =
Files.readAllLines(pathFileToRead);
                //System.out.println(lines);

                Files.lines(pathFileToRead)
                        .map(String::toLowerCase)
                        .filter(str -> str.contains("a"))
                        .forEach(System.out::println);

        }

}
```

/17-Files/src/files/FileWriteRunner.java

```
package files;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
```

```
import java.util.List;

public class FileWriteRunner {



        public static void main(String[] args) throws
IOException {
                Path pathFileToWrite =
Paths.get("./resources/file-write.txt");

                List<String> list  =
                                        List.of("Apple",
"Boy", "Cat", "Dog", "Elephant");

                Files.write(pathFileToWrite, list);


        }

}
```

/17-Files/resources/data.txt

```
123,122
jljlfsadf
Apple
Bat
Cat
```

/17-Files/resources/file-write.txt

```
Apple
Boy
```

```
Cat Dog

Elephant
```

More Concurrency - Concurrent Collections & More..

Steps

- Step 01 - Getting started with Synchronized
- Step 02 - Problem with Synchronized - Less Concurrency
- Step 03 - Enter Locks with ReEntrantLock
- Step 04 - Introduction to Atomic Classes - AtomicInteger
- Step 05 - Need for ConcurrentMap
- Step 06 - Implementing an example with ConcurrentHashMap
- Step 07 - ConcurrentHashMap uses different locks for diferrent regions
- Step 08 - CopyOnWrite Concurrent Collections - When reads are more than writes
- Step 09 - Conclusion

Code Examples

/src/com/in28minutes/concurrency/BiCounter.java

```
package com.in28minutes.concurrency;

public class BiCounter {
  private int i = 0;
  private int j = 0;

  synchronized public void incrementI() {
    i++;
    //get
```

```
 i
    //increment
    //set i
  }

  public int getI() {
    return i;
   }

  synchronized public void incrementJ() {
    j++;
    //get j
    //increment
    //set j
  }

  public int getJ() {
    return j;
  }

}
```

/src/com/in28minutes/concurrency/BiCounterWithAtomicInteger.java

```
package com.in28minutes.concurrency;

import java.util.concurrent.atomic.AtomicInteger;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class BiCounterWithAtomicInteger {
  private AtomicInteger i = new AtomicInteger();
  private AtomicInteger j = new
```

```
 AtomicInteger();

  public void incrementI() {
    i.incrementAndGet();
  }

  public int getI() {
    return  i.get();
  }

  public void incrementJ() {
    j.incrementAndGet();
  }

  public int getJ() {
    return j.get();
  }

}
```

/src/com/in28minutes/concurrency/BiCounterWithLock.java

```
package com.in28minutes.concurrency;

import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class BiCounterWithLock {
  private int i = 0;
  private int j = 0;

  Lock lockForI = new ReentrantLock();
  Lock lockForJ = new ReentrantLock();

  public void incrementI() {
    lockForI.lock();//Get Lock For
```

```
 I
    i++;
    lockForI.unlock();
    //Release Lock For I
   }

  public int getI() {
    return i;
  }

  public void incrementJ() {
    lockForJ.lock();//Get Lock For J
    j++;
    lockForJ.unlock();//Release Lock For J
  }

  public int getJ() {
    return j;
  }

}
```

/src/com/in28minutes/concurrency/ConcurrencyRunner.java

```
package com.in28minutes.concurrency;

public class ConcurrencyRunner {

  public static void main(String[] args) {
    Counter counter = new Counter();
    counter.increment();
    counter.increment();
    counter.increment();
```

```
 System.out.println(counter.getI());

  }

}
```

/src/com/in28minutes/concurrency/ConcurrentMapRunner.java

```
package com.in28minutes.concurrency;

import java.util.Hashtable;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentMap;
import java.util.concurrent.atomic.LongAdder;

public class ConcurrentMapRunner {

  public static void main(String[] args) {
    ConcurrentMap<Character, LongAdder> occurances = new
ConcurrentHashMap<>();

    String str = "ABCD ABCD ABCD";

    for(char character:str.toCharArray()) {
      occurances.computeIfAbsent(character, ch -> new
LongAdder()).increment();
    }

    System.out.println(occurances);


  }
```

```
 /*
      Map<Character, LongAdder> occurances = new

 Hashtable<>();

    String str = "ABCD ABCD  ABCD";
    for(char character:str.toCharArray()) {
      LongAdder longAdder = occurances.get(character);
      if(longAdder == null) {
        longAdder = new LongAdder();
      }
      longAdder.increment();
      occurances.put(character, longAdder);
    }

    System.out.println(occurances);

   */
}
```

/src/com/in28minutes/concurrency/CopyOnWriteArrayListRunner.java

```
package com.in28minutes.concurrency;

import java.util.List;
import java.util.concurrent.CopyOnWriteArrayList;

public class CopyOnWriteArrayListRunner {

  //add - copy
  //get

  public static void main(String[] args) {
    List<String> list = new
```

```
 CopyOnWriteArrayList<>();

    //Threads - 3


 list.add("Ant");
     list.add("Bat");
    list.add("Cat");

    //Threads - 10000
    for(String element:list) {
      System.out.println(element);
    }

    // TODO Auto-generated method stub

  }

}
```

/src/com/in28minutes/concurrency/Counter.java

```
package com.in28minutes.concurrency;

public class Counter {
  private int i = 0;

  synchronized public void increment() {
    i++;
    //get i
    //increment
    //set i
  }

  public int getI() {
    return
```

```
 i;
  }
}
```

Java Tips

Steps

- Java Tip 01 - Imports and Static Imports
- Java Tip 02 - Blocks
- Java Tip 03 - equals method
- Java Tip 04 - hashcode method
- Java Tip 05 - Class Access Modifiers - public and default
- Java Tip 06 - Method Access Modifiers - public, protected, private and default
- Java Tip 07 - Final classes and Final methods
- Java Tip 08 - Final Variables and Final Arguments
- Java Tip 09 - Why do we need static variables?
- Java Tip 09 - Why do we need static methods?
- Java Tip 10 - Static methods cannot use instance methods or variables
- Java Tip 11 - public static final - Constants
- Java Tip 12 - Nested Classes - Inner Class vs Static Nested Class
- Java Tip 13 - Anonymous Classes
- Java Tip 14 - Why Enum and Enum Basics - ordinal and values
- Java Tip 15 - Enum - Constructor, variables and methods
- Java Tip 16 - Quick look at inbuild Enums - Month, DayOfWeek

Code Examples

/src/com/in28minutes/tips/access/package1/ClassAccessModifiers.java

```
package com.in28minutes.tips.access.package1;
```

```
//public, protected, (default), private
public class ClassAccessModifiers {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    ClassAccessModifiers c = new ClassAccessModifiers();

  }

}
```

/src/com/in28minutes/tips/access/package1/ExampleClass.java

```
package com.in28minutes.tips.access.package1;

public class ExampleClass {
  public void publicMethod() {}
  protected void protectedMethod() {}
  private void privateMethod() {}
  void defaultMethod() {}

  public static void main(String[] args) {
    ExampleClass exampleClass = new ExampleClass();
    exampleClass.privateMethod();
    exampleClass.protectedMethod();
    exampleClass.publicMethod();
    exampleClass.defaultMethod();
  }
}
```

/src/com/in28minutes/tips/access/package1/MethodAccessRunnerInsideSamePackage.java

```
package com.in28minutes.tips.access.package1;

public class MethodAccessRunnerInsideSamePackage {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    ExampleClass exampleClass = new ExampleClass();
    //exampleClass.privateMethod();
    exampleClass.protectedMethod();
    exampleClass.publicMethod();
    exampleClass.defaultMethod();

  }

}
```

/src/com/in28minutes/tips/access/package2/ClassAccessModifiersRunnerInOtherPackage.java

```
package com.in28minutes.tips.access.package2;

import
com.in28minutes.tips.access.package1.ClassAccessModifiers;

//public, protected, (default), private
public class ClassAccessModifiersRunnerInOtherPackage {

  public static void main(String[] args) {
    ClassAccessModifiers c = new ClassAccessModifiers();
  }

}
```

/src/com/in28minutes/tips/access/package2/MethodAccessRunnerInDifferentPackage.java

```
package com.in28minutes.tips.access.package2;

import com.in28minutes.tips.access.package1.ExampleClass;

public class MethodAccessRunnerInDifferentPackage {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    ExampleClass exampleClass = new ExampleClass();

    //exampleClass.privateMethod();
    //exampleClass.protectedMethod();
    exampleClass.publicMethod();
    //exampleClass.defaultMethod();

  }

}
```

/src/com/in28minutes/tips/anonymous/AnonymousClassRunner.java

```
package com.in28minutes.tips.anonymous;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class AnonymousClassRunner
```

```
 {

  public static void main(String[] args) {
    List<String> animals = new ArrayList<String>(
        List.of("Ant", "Cat", "Ball", "Elephant"));

    Comparator<String> lengthComparator = new
Comparator<String>() {
      @Override
      public int compare(String str1, String str2) {
        return Integer.compare(str1.length(),
str2.length());
      }
    };

    Collections.sort(animals,
        lengthComparator
    );
    System.out.println(animals);

  }

}
```

/src/com/in28minutes/tips/blocks/BlocksRunner.java

```
package com.in28minutes.tips.blocks;

public class BlocksRunner {
  public static final int SECONDS_IN_MINUTE = 60;
  public static final int MINUTES_IN_HOUR = 60;
  public static final int HOURS_IN_DAY = 24;
  public static final int SECONDS_IN_DAY
    = SECONDS_IN_MINUTE * MINUTES_IN_HOUR * HOURS_IN_DAY;

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

```
 {
    //System.out.print(Integer.MIN_VALUE);
    System.out.print(SECONDS_IN_DAY);

    //System.out.print("main");

    {
      int i;
      //System.out.print("3>2");
        //System.out.print("3>2");
    }

    //System.out.print(i);

  }

}
```

/src/com/in28minutes/tips/eclipse/DummyForTest.java

```
package com.in28minutes.tips.eclipse;

import java.util.ArrayList;
import java.util.List;

public class DummyForTest {

  public void doSomething() {
    List list = new ArrayList();
  }

}
```

/src/com/in28minutes/tips/eclipse/EclipseTipsAndTricks.java

```
package com.in28minutes.tips.eclipse;

import java.math.BigDecimal;

//PAIR PROGRAMMING

class TestBean {
  private int i; //i is awesome
  private String str;



  public TestBean() {
    /*
     fsadjflkas
     fskljdfalsk
     */
    super();
  }

  public TestBean(int i, String str) {
    super();
    this.i = i;
    this.str = str;
  }
  /* (non-Javadoc)
   * @see java.lang.Object#hashCode()
   */
  @Override
  public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + i;
    return result;
  }
  /*
```

```
(non-Javadoc)
  * @see java.lang.Object#equals(java.lang.Object)
  */
 @Override
 public boolean equals(Object obj) {
   if (this == obj) {
     return true;
   }
   if (obj == null) {
     return false;
   }
   if (getClass() != obj.getClass()) {
     return false;
   }
   TestBean other = (TestBean) obj;
   if (i != other.i) {
     return false;
   }
   return true;
 }
 /**
  * @return the i
  */
 public int getI() {
   return i;
 }
 /**
  * @param i the i to set
  */
 public void setI(int i) {
   this.i = i;
 }
 /**
  * @return the str
  */
 public String getStr()
```

```
 {
    return str;
  }
  /**
   * @param str the str to set
   */
  public void setStr(String str) {
    this.str = str;
  }



}

class Test implements Comparable<String> {

  @Override
  public int compareTo(String arg0) {
    return 0;
  }

}

public class EclipseTipsAndTricks {

  public static void main(String[] args) throws
InterruptedException {

    DummyForTest dt = new DummyForTest();
    dt.doSomething();
    BigDecimal bd = new BigDecimal(25);
    Thread.sleep(returnSomething());
  }

  private static int returnSomething() {
    return 1000 * 45 *
```

```
 456;
  }

}
```

/src/com/in28minutes/tips/enums/EnumRunner.java

```
package com.in28minutes.tips.enums;

import java.util.Arrays;

public class EnumRunner {

  public static void main(String[] args) {
    Season season = Season.FALL;

    Season season1 = Season.valueOf("WINTER");
    System.out.println(season1);
    System.out.println(Season.SPRING.ordinal());
    System.out.println(Season.SPRING.getValue());

    System.out
        .println(Arrays.toString(Season.values()));

  }

}
```

/src/com/in28minutes/tips/enums/Season.java

```
package com.in28minutes.tips.enums;
public enum Season {
  SPRING(4), SUMMER(1), WINTER(2), FALL(3) ;

  private int
```

```
 value;

  private Season(int value) {
    this.value = value;
  }

  public int getValue() {
    return value;
  }


  //0,1,2,3
}
```

/src/com/in28minutes/tips/equals/EqualsRunner.java

```
package com.in28minutes.tips.equals;

class Client {
  private int id;

  public Client(int id) {
    super();
    this.id = id;
  }

  @Override
  public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + id;
    return result;
  }

  @Override
  public boolean equals(Object that)
```

```
 {
    if (this == that)
      return true;

    if (that == null)
      return false;
    if (getClass() != that.getClass())
      return false;
    Client other = (Client) that;
    if (id != other.id)
      return false;
    return true;
  }

  //equals
  //hashcode



}

public class EqualsRunner {

  public static void main(String[] args) {
    Client c1 = new Client(1);
    Client c2 = new Client(1);
    Client c3 = new Client(2);
    System.out.println(c1.equals(c2));
    System.out.println(c1.equals(c3));


  }

}
```

/src/com/in28minutes/tips/imports/ImportsRunner.java

```
package com.in28minutes.tips.imports;

//import java.lang.*; //DEFAULT
import java.math.BigDecimal;
import java.util.ArrayList;
//import java.util.Collections;

import static java.lang.System.out;
import static java.util.Collections.*;

public class ImportsRunner {

  public static void main(String[] args) {

    out.println("IMports");
    out.println("Static Imports");

    sort(new ArrayList<String>());

    BigDecimal db = null;

  }

}
```

/src/com/in28minutes/tips/nonaccess/package1/FinalNonAccessModifierRunner.java

```
package com.in28minutes.tips.nonaccess.package1;

final class FinalClass {

}

//class SomeClass extends FinalClass{
```

```
//}

class SomeClass {
  final public void doSomething() {}
  public void doSomethingElse(final int arg) {
    //arg = 6;

  }
}

class ExtendingClass extends SomeClass {
  //public void doSomething() {}
}

public class FinalNonAccessModifierRunner {

  public static void main(String[] args) {
    final int i;
    i=5;

    //i = 7;

  }

}
```

/src/com/in28minutes/tips/nonaccess/package1/StaticModifierRunner.java

```
package com.in28minutes.tips.nonaccess.package1;

class Player{
  private String name;

  private static int count =
```

```
 0;

  public Player(String name) {
    super();
    this.name = name;
    count++;
  }

  static public int getCount() {
    return count;
  }

  public String getName() {
    System.out.println(count);
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }


}

public class StaticModifierRunner {

  public static void main(String[] args) {
    Player player1 = new Player("Ronaldo");

    System.out.println(Player.getCount());

    Player player2 = new Player("Federer");

    System.out.println(Player.getCount());
  }
```

```
}
```

in28minutes

Become an expert on Java, Spring Boot, APIs, Microservices and Full Stack Development

Checkout the Complete in28Minutes Course Guide