CORE JAVA

KavinSchool

ARRAYS IN JAVA

KavinSchool

1

INTRODUCTION TO

ARRAYS

KavinSchool

Learning Objectives

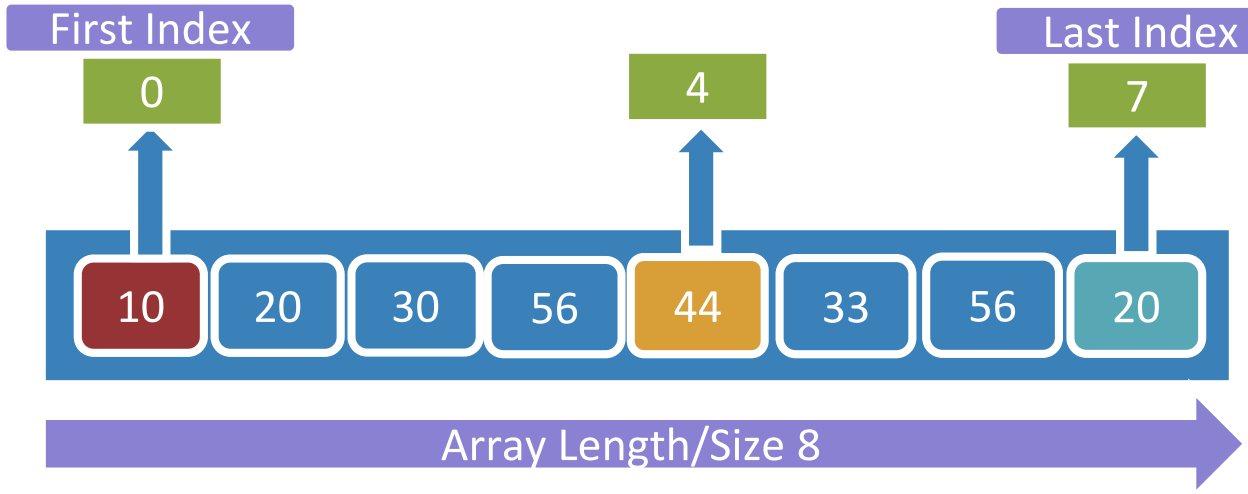
- What are Arrays?
- Pros/Cons
- Single Dim Arrays
- 2d Arrays
- 3d Arrays

ARRAYS

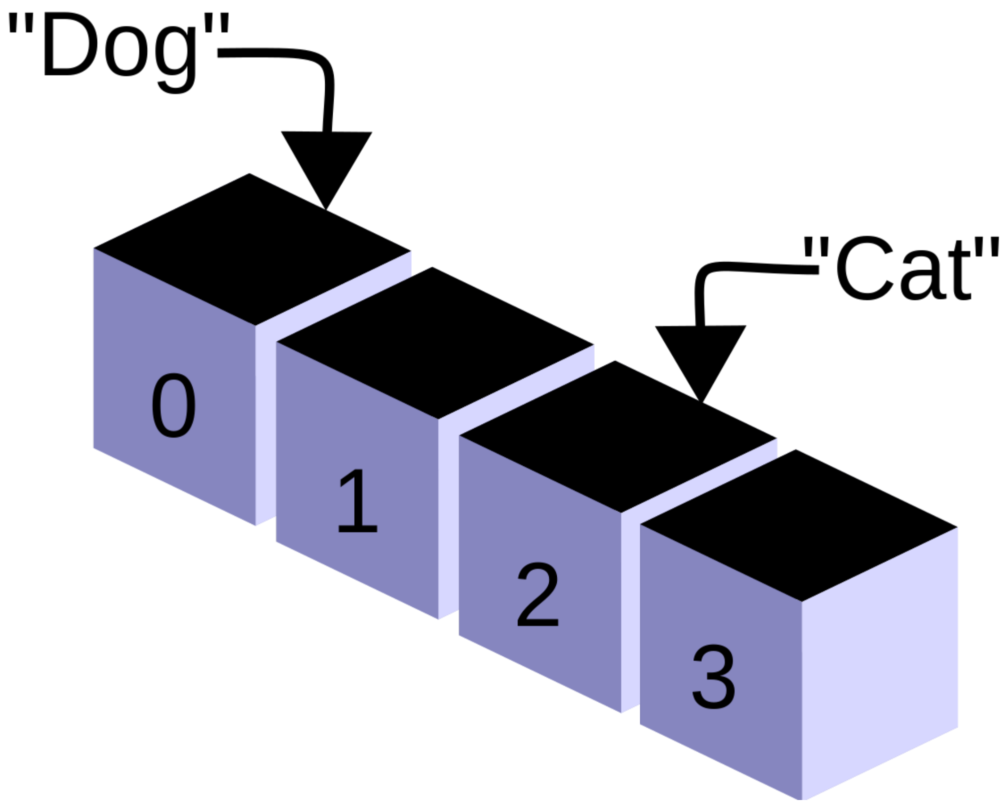
- An array is a common type of data structure where all elements must be of the same data type.
- In Java, once defined, the size of an array is fixed and cannot increase to accommodate more elements. The first element of an array starts with index zero.

ARRAYS

- An array is a common type of data structure where all elements must be of the same data type.
- In Java, once defined, the size of an array is fixed and cannot increase to accommodate more elements. The first element of an array starts with index zero.



ARRAY INDEXES



PROS/CONS OF ARRAYS

- Pros:
  - **Code Optimization:** You can retrieve or sort the data easily
  - **Random access:** You can get any data located at any index position without going through them sequentially
- Cons:
  - **Size Limit:** You can store only fixed size of elements in the array. Arrays does not grow its size at runtime.
  - You can use collection framework where ArrayList allows you to grow the Arrays without Size limitation.

ARRAYS

An array is a common type of data structure where all elements must be of the same data type.

In Java, the first element of an array starts with an index value of zero.

TYPES OF ARRAYS

- Single Dimensional Arrays
- Multi Dimensional Arrays

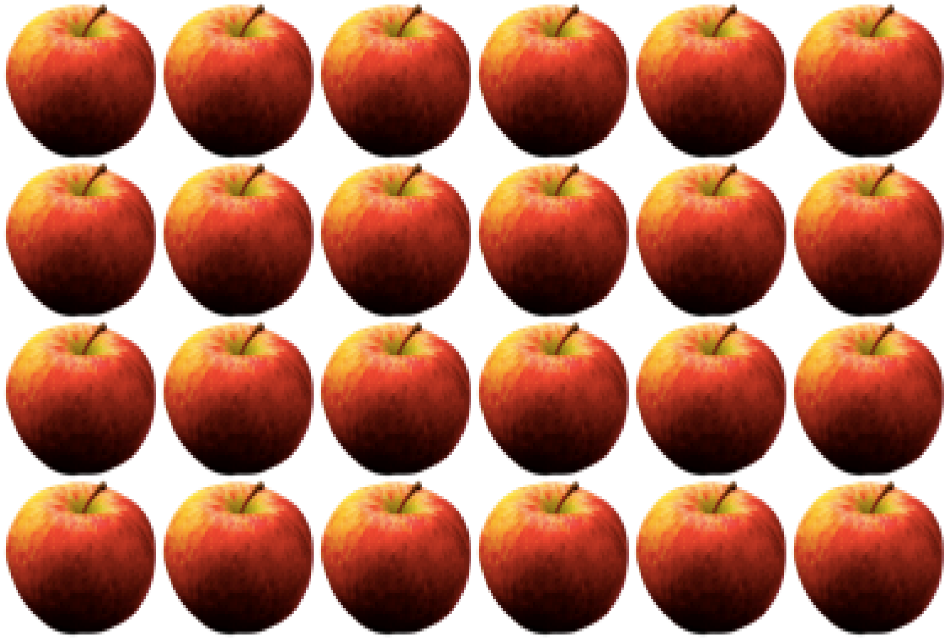
Single Dim Syntax

```
dataType[] arr;
dataType []arr;

dataType arr[];
```

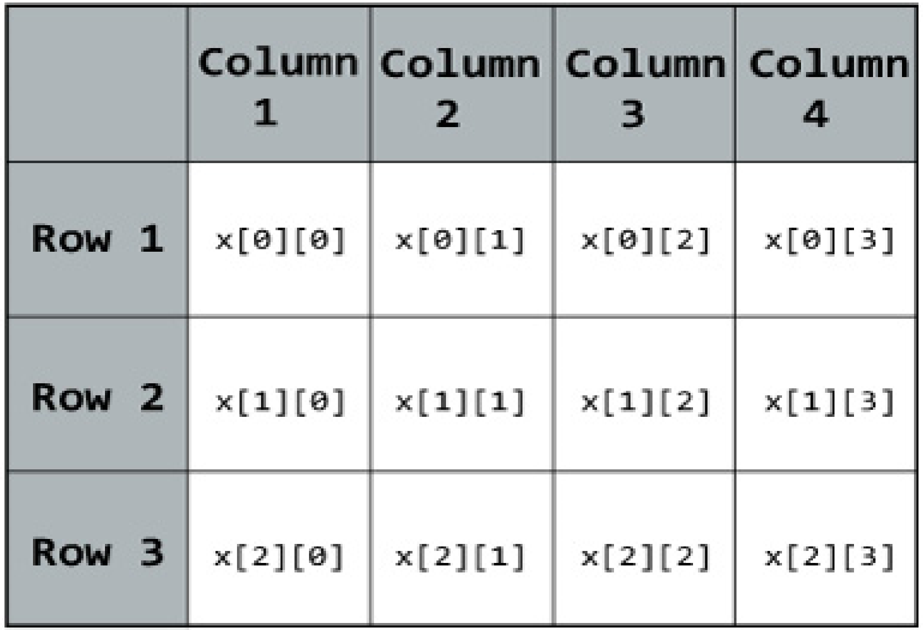
Multi Dim Syntax

```
dataType[][] arrayRefVar;
dataType [][]arrayRefVar;

dataType arrayRefVar[][];
dataType []arrayRefVar[];
```

ARRAY OF APPLES



2D ARRAYS



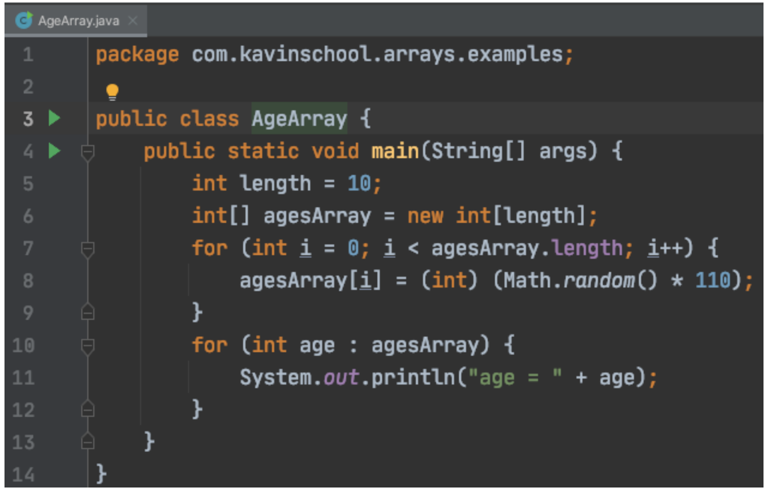
	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

SINGLE DIM ARRAYS

➢ Single Dimensional Arrays Example

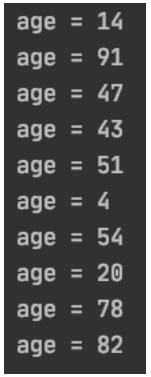
Code

```
int length = 10;
int[] agesArray = new int[length];
for (int i = 0; i < agesArray.length; i++) {
    agesArray[i] = (int) (Math.random() * 110);
}

for (int age : agesArray) {
    System.out.println("age = " + age);
}
```



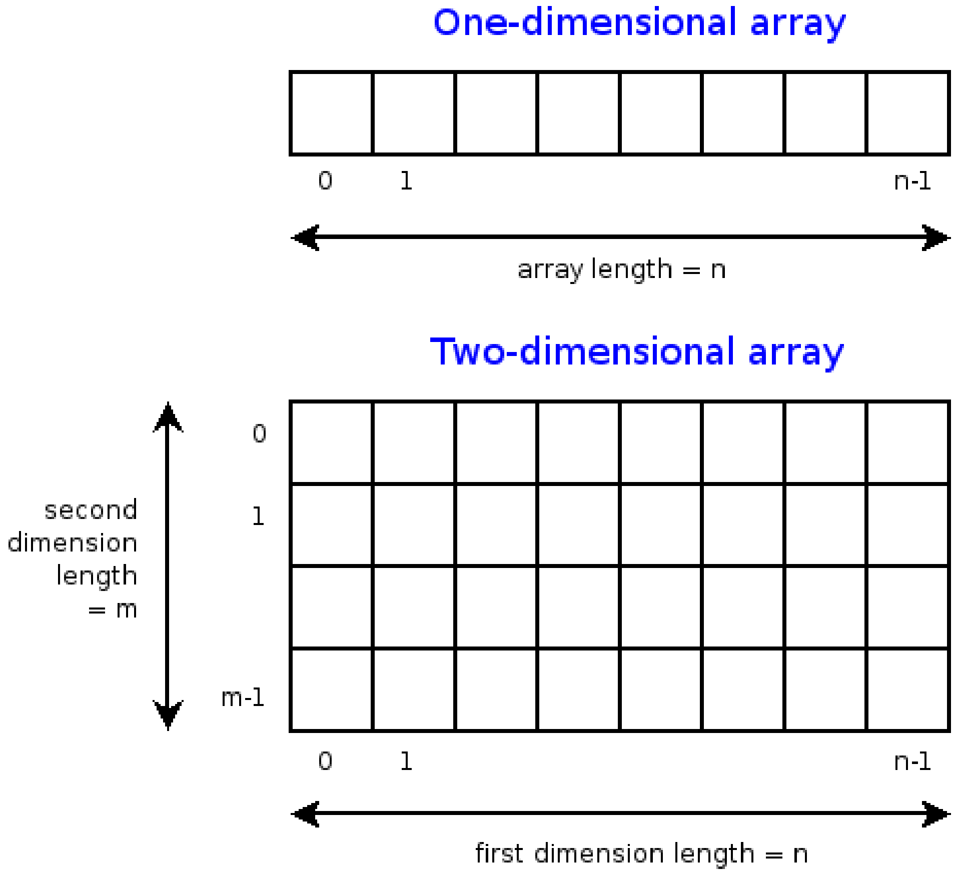
AgeArray.java

```
package com.kavinschool.arrays.examples;

public class AgeArray {
    public static void main(String[] args) {
        int length = 10;
        int[] agesArray = new int[length];
        for (int i = 0; i < agesArray.length; i++) {
            agesArray[i] = (int) (Math.random() * 110);
        }
        for (int age : agesArray) {
            System.out.println("age = " + age);
        }
    }
}
```



```
age = 14
age = 91
age = 47
age = 43
age = 51
age = 4
age = 54
age = 20
age = 78
age = 82
```

2D VS 1D

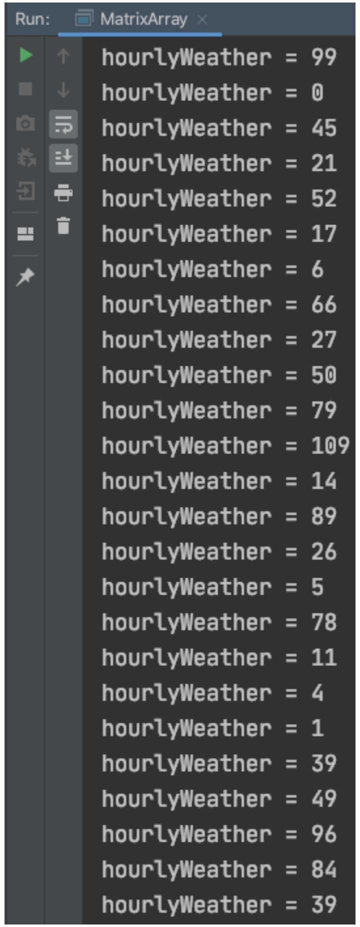

One-dimensional array
0
1
n-1
array length = n
Two-dimensional array
second dimension length = m
0
1
m-1
0
1
n-1
first dimension length = n

2 DIM ARRAYS

➢ Two-Dimensional Arrays Example

Code

```
int[][] WeeksWeather = new int[7][24];

for (int i = 0; i < 7; i++) {
    for (int j = 0; j < 24; j++) {
        WeeksWeather[i][j] = (int) (Math.random() * 110);
    }
}

for (int[] dailyWeather : WeeksWeather) {
    for (int hourlyWeather : dailyWeather) {
        System.out.println("hourlyWeather = " + hourlyWeather);
    }
}
```


Run: MatrixArray

```
hourlyWeather = 99
hourlyWeather = 0
hourlyWeather = 45
hourlyWeather = 21
hourlyWeather = 52
hourlyWeather = 17
hourlyWeather = 6
hourlyWeather = 66
hourlyWeather = 27
hourlyWeather = 50
hourlyWeather = 79
hourlyWeather = 109
hourlyWeather = 14
hourlyWeather = 89
hourlyWeather = 26
hourlyWeather = 5
hourlyWeather = 78
hourlyWeather = 11
hourlyWeather = 4
hourlyWeather = 1
hourlyWeather = 39
hourlyWeather = 49
hourlyWeather = 96
hourlyWeather = 84
hourlyWeather = 39
```

3 DIM ARRAYS

- Three Dim Arrays Example

Code

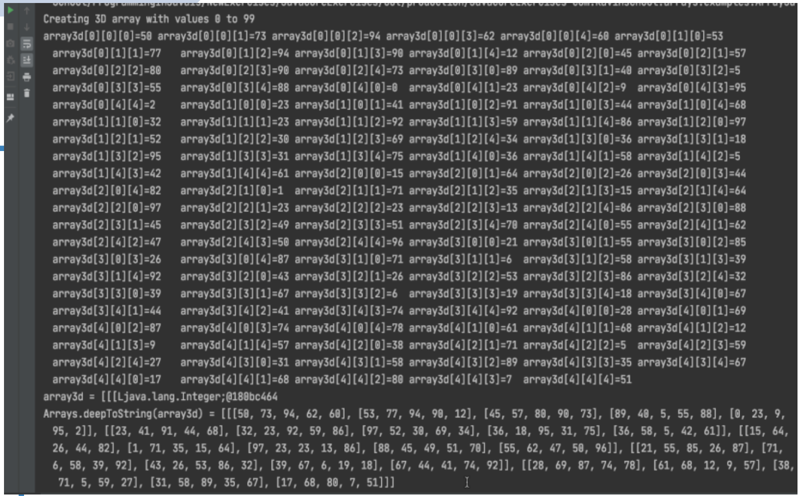
```
Integer[][][] array3d = new Integer[5][5][5];
//Store in a 3d array
System.out.println("Creating 3D array with values 0 to 99");
for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
        for (int k = 0; k < 5; k++) {
            array3d[i][j][k] = (int) (Math.random() * 100);
            System.out.printf("array3d[%d][%d][%d]=%d\t", i,j,k,array3d[i][j][k]);
        }
    }
}
System.out.println("\narray3d = " + array3d);
System.out.println("Arrays.deepToString(array3d) = " +
Arrays.deepToString(array3d));
```

3 DIM ARRAYS

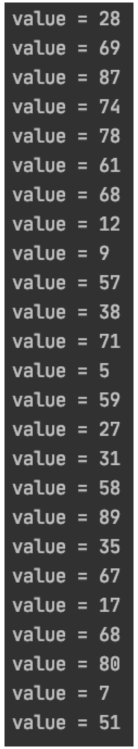
➢ Three Dim Arrays Example

Code

```java
System.out.println("Other ways to access 3D array");
for (int i = 0; i < array3d.length; i++) {
    Integer[][] array2d = array3d[i];
    for (int j = 0; j < array2d.length; j++) {
        Integer[] array1d = array2d[j];
        for (int k = 0; k < array1d.length; k++) {
            Integer value = array1d[k];
            System.out.println("value = " + value);
        }
    }
}
```



```
Creating 3D array with values 0 to 99
array3d[0][0][0]=50 array3d[0][0][1]=73 array3d[0][0][2]=94 array3d[0][0][3]=62 array3d[0][0][4]=60 array3d[0][1][0]=53
  array3d[0][1][1]=77   array3d[0][1][2]=94 array3d[0][1][3]=90 array3d[0][1][4]=12 array3d[0][2][0]=45 array3d[0][2][1]=57
  array3d[0][2][2]=80   array3d[0][2][3]=90 array3d[0][2][4]=73 array3d[0][3][0]=89 array3d[0][3][1]=40 array3d[0][3][2]=5
  array3d[0][3][3]=55   array3d[0][3][4]=88 array3d[0][4][0]=0  array3d[0][4][1]=23 array3d[0][4][2]=9  array3d[0][4][3]=95
  array3d[0][4][4]=2    array3d[1][0][0]=23 array3d[1][0][1]=41 array3d[1][0][2]=91 array3d[1][0][3]=44 array3d[1][0][4]=68
  array3d[1][1][0]=32   array3d[1][1][1]=23 array3d[1][1][2]=92 array3d[1][1][3]=59 array3d[1][1][4]=86 array3d[1][2][0]=97
  array3d[1][2][1]=52   array3d[1][2][2]=30 array3d[1][2][3]=69 array3d[1][2][4]=34 array3d[1][3][0]=36 array3d[1][3][1]=18
  array3d[1][3][2]=95   array3d[1][3][3]=31 array3d[1][3][4]=75 array3d[1][4][0]=36 array3d[1][4][1]=58 array3d[1][4][2]=5
  array3d[1][4][3]=42   array3d[1][4][4]=61 array3d[2][0][0]=15 array3d[2][0][1]=64 array3d[2][0][2]=26 array3d[2][0][3]=44
  array3d[2][0][4]=82   array3d[2][1][0]=1  array3d[2][1][1]=71 array3d[2][1][2]=35 array3d[2][1][3]=15 array3d[2][1][4]=64
  array3d[2][2][0]=97   array3d[2][2][1]=23 array3d[2][2][2]=23 array3d[2][2][3]=13 array3d[2][2][4]=86 array3d[2][3][0]=88
  array3d[2][3][1]=45   array3d[2][3][2]=49 array3d[2][3][3]=51 array3d[2][3][4]=70 array3d[2][4][0]=55 array3d[2][4][1]=62
  array3d[2][4][2]=47   array3d[2][4][3]=50 array3d[2][4][4]=96 array3d[3][0][0]=21 array3d[3][0][1]=55 array3d[3][0][2]=85
  array3d[3][0][3]=26   array3d[3][0][4]=87 array3d[3][1][0]=71 array3d[3][1][1]=6  array3d[3][1][2]=58 array3d[3][1][3]=39
  array3d[3][1][4]=92   array3d[3][2][0]=43 array3d[3][2][1]=26 array3d[3][2][2]=53 array3d[3][2][3]=86 array3d[3][2][4]=32
  array3d[3][3][0]=39   array3d[3][3][1]=67 array3d[3][3][2]=6  array3d[3][3][3]=19 array3d[3][3][4]=18 array3d[3][4][0]=67
  array3d[3][4][1]=44   array3d[3][4][2]=41 array3d[3][4][3]=74 array3d[3][4][4]=92 array3d[4][0][0]=28 array3d[4][0][1]=69
  array3d[4][0][2]=87   array3d[4][0][3]=74 array3d[4][0][4]=78 array3d[4][1][0]=61 array3d[4][1][1]=68 array3d[4][1][2]=12
  array3d[4][1][3]=9    array3d[4][1][4]=57 array3d[4][2][0]=38 array3d[4][2][1]=71 array3d[4][2][2]=5  array3d[4][2][3]=59
  array3d[4][2][4]=27   array3d[4][3][0]=31 array3d[4][3][1]=58 array3d[4][3][2]=89 array3d[4][3][3]=35 array3d[4][3][4]=67
  array3d[4][4][0]=17   array3d[4][4][1]=68 array3d[4][4][2]=80 array3d[4][4][3]=7  array3d[4][4][4]=51
array3d = [[[Ljava.lang.Integer;@180bc464
Arrays.deepToString(array3d) = [[[50, 73, 94, 62, 60], [53, 77, 94, 90, 12], [45, 57, 80, 90, 73], [89, 40, 5, 55, 88], [0, 23, 9,
  95, 2]], [[23, 41, 91, 44, 68], [32, 23, 92, 59, 86], [97, 52, 30, 69, 34], [36, 18, 95, 31, 75], [36, 58, 5, 42, 61]], [[15, 64,
  26, 44, 82], [1, 71, 35, 15, 64], [97, 23, 23, 13, 86], [88, 45, 49, 51, 70], [55, 62, 47, 50, 96]], [[21, 55, 85, 26, 87], [71,
  6, 58, 39, 92], [43, 26, 53, 86, 32], [39, 67, 6, 19, 18], [67, 44, 41, 74, 92]], [[28, 69, 87, 74, 78], [61, 68, 12, 9, 57], [38,
   71, 5, 59, 27], [31, 58, 89, 35, 67], [17, 68, 80, 7, 51]]]
```



```
value = 28
value = 69
value = 87
value = 74
value = 78
value = 61
value = 68
value = 12
value = 9
value = 57
value = 38
value = 71
value = 5
value = 59
value = 27
value = 31
value = 58
value = 89
value = 35
value = 67
value = 17
value = 68
value = 80
value = 7
value = 51
```

KavinSchool

THANKS!

Your feedback is welcome
support@kavinschool.com

ENUM IN JAVA

KavinSchool

2

INTRODUCTION TO

ENUMURATION

KavinSchool

Learning Objectives

- What are Enums?
- Pros/Cons
- Examples
- Usages

WHAT IS ENUM?

- An enum type is a special data type that enables for a variable to be a set of predefined constants.
- Instead of using arbitrary numbers, enum allows to use permissible items in a construct called enumeration
- An enumeration is a special class that provides a type-safe implementation of constant data for your program

An enum can be used to define set of enum constants.

enum Syntax

```
enum {
    CONST1,
    CONST2,
    CONST3
}
```

ENUM - EXAMPLE

- An enum is a reference data type that holds a reference to memory in the heap (like class, interface and array).
- The constants are implicitly static final, the values can not be modified.
- Enum is type-safe
- Any static constants you can refer them with their own name space.

Code

```
public enum Suit {
                SPADE,
                DIAMOND,
                CLUB,
                HEART
}
```

PROS/CONS OF ENUMS

- Pros:
  - Type-safe
  - Have own namespace
  - All constants are public static final
  - Can be accessed via EnumName.ConstantName
  - Can be used with switch statements in place of int
- Cons
  - Can not extend another class or enum
  - Can not instantiate an enum

ENUM WEEKDAYS

- Normal order of enum's starts from 0

Code

```
public enum WeekDays {
   //Normal order MONDAY=0, TUESDAY=1
   MONDAY,
   TUESDAY,
   WEDNESDAY,
   THURSDAY,
   FRIDAY,
   SATURDAY,
   SUNDAY;
}
```

USE WITH SWITCH

- You can enum with switch as you use with int

Code

```
switch (whichDay) {
   case MONDAY: System.out.println("Monday");
   case TUESDAY: System.out.println("Tuesday");
   case THURSDAY: System.out.println("Thursday");
   case SATURDAY: System.out.println("Saturday");
      System.out.println(" Vegitarian");
      break;
   case WEDNESDAY: System.out.println("Wednesday");
   case FRIDAY: System.out.println("Friday");
   case SUNDAY: System.out.println("Sunday");
      System.out.println(" Non Vegitarian");
      break;
   default: System.out.println("Are you on earth?");
      break;
}
```

ENUM WITH VALUES

- You can assign a value to your enum constants

Code

```
public enum NewWeekDays {
   //Assign specific value for your constants
   MONDAY(1),
   TUESDAY(2),
   WEDNESDAY(3),
   THURSDAY(4),
   FRIDAY(5),
   SATURDAY(6),
   SUNDAY(7);

   int day;

   NewWeekDays(int whichDay) {
      day = whichDay;
   }

   public int getDay() {
      return day;
   }
}
```

ENUM WITH METHODS

- You can define an abstract method with enum constants
- An enum type can have abstract methods just like a class
- Each enum constant needs to implement the abstract method

Code

```
public enum TrafficSignal {
   RED(40) {
      public TrafficSignal next() {
         return GREEN;
      }
   }, YELLOW(10) {
      public TrafficSignal next() {
         return RED;
      }
   }, GREEN(30) {
      public TrafficSignal next() {
         return YELLOW;
      } };
    public abstract TrafficSignal next();
```

ENUM WITH METHODS

- You can define an abstract method with enum constants

Code

```
private final int seconds;

    TrafficSignal(int seconds) {
        this.seconds = seconds;
    }

    int getSeconds() {
        return seconds;
    }

    public static void main(String[] args) {
        for (TrafficSignal light : TrafficSignal.values()) {
            System.out.printf("%s: %d seconds, next is %s\n", light,
                    light.getSeconds(), light.next());
        }
    }
}
```

ENUM SPECIAL METHODS

- You can use any of the below methods with enums

Code

```
public final String name();
// Returns the name of the constant as defined in the declaration.
// You could also override the toString() to provide a more user-friendly description of your
constant name

public final int ordinal();
// Returns the position value starting from 0.

public static <T extends Enum<T>> T valueOf(Class<T> enumType,String name);
// This method Returns the enum constant of this type with the specified name. The string must
match exactly an identifier used to declare an enum constant in this type
```

ENUM SPECIAL METHODS

- You can use any of the below methods with enums

```
public class GenericExtends {

        enum Day {
                MONDAY, TUESDAY, WEDNESDAY;
        }

        public static void main(String[] args) {
                Day day = Enum.valueOf(Day.class, "MONDAY");
                System.out.println(day); // Outputs: MONDAY
        }

}
```

Enum.valueOf uses <T extends Enum<T>> to ensure that only Day (an enum) can be passed as the type argument. This provides type safety and proper enum handling without allowing an enum to inherit from another type

ENUM – VALUES(), VALUESOF()

- values() method can be used to return all values present inside enum
- valueOf() method returns the enum constant of the specified string value if exists
- ordinal() method returns each enum constant index
- name() methods return the name of the enum constant as it is declared which can be overriden using to toString() method

Code

```
HandSignal player1HandSignal = HandSignal.values()[player1];
HandSignal player2HandSignal = HandSignal.values()[player2];
System.out.println(TrafficSignal.valueOf("GREEN"));
System.out.println(TrafficSignal.valueOf("RED"));
TrafficSignal yellow = TrafficSignal.valueOf("YELLOW");
System.out.println(yellow.nextSignal());
System.out.println(yellow.ordinal());
System.out.println(yellow.name());
```

ENUM – CONSTRUCTOR

- The enum can contain constructor and it is executed separately for each enum constant at the time of enum class loading
- The constructor for an enum type must be package-private or private access.
- The constructor automatically creates the constants that are defined at the beginning of the enum body. You cannot invoke an enum constructor yourself

Code

```
TrafficSignal(int seconds) {
        this.seconds = seconds;
   }
```

THANKS!

Your feedback is welcome
support@kavinschool.com

3

LESSON

EXTERNAL PACKAGES

WORKING WITH EXTERNAL PACKAGES

- Package names allow us to separate the code base without any conflicts
- Java classes must specify their package before the class definition, else the default package is assumed
- To define a package name, use the reverse domain name followed by your project name
  - com.kavinschool.payroll.system
  - com.kavinschool.oops.examples

IMPORTS

- To refer to any class you need a qualified reference to its package
- import statement allows to resolve class references
- For example, to use SecureRandom methods, we can import as follow
    - **import java.security.SecureRandom**