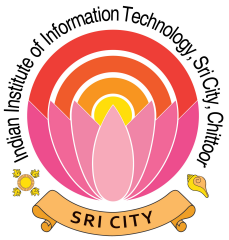


Monsoon 2019

Keywords – this and static

Object Oriented Programming

by

Dr. Rajendra Prasath

Indian Institute of Information Technology

Sri City – 517 646, Andhra Pradesh, India

Recap: Objects in JAVA ?

- An entity that has **state** and **behaviour** is known as an object
  - **Examples:** Chair, bike, marker, pen, table, car etc
  - It can be physical or logical
- An object has three characteristics:
  - **State:** represents data (value) of an object
  - **Behaviour:** represents the behaviour (functionality) of an object such as deposit, withdraw and so on
  - **Identity (Internally used):**
    - Signature (unique) of the object
    - Object identity is typically implemented via a unique ID
    - The value of the ID is not visible to the external user
    - But, Internally by JVM to identify each object uniquely

Recap: Method Overloading

Whenever same method name is exiting multiple times in the same class with different number of parameter or different order of parameters or different types of parameters is known as method overloading

Why use Method Overloading in Java ?

- Suppose we have to perform addition of given number but there can be any number of arguments, if we write method such as a(int, int)for two arguments, b(int, int, int) for three arguments then it is very difficult for you and other programmer to understand purpose or behaviors of method they can not identify purpose of method.
- So use method overloading
- Example: Write
  - sum(int, int) for two arguments
  - sum(int, int, int) using method overloading concept.

Recap: Exercise - 3

- **Practice Problems using various access modifiers**

  - Apply it for doing Basic Arithmetic operations
    - Addition
    - subtraction
    - Multiplication
    - Division
    - Modular division
    - Also mathematical functions

  - Develop an application **Basic Calculator** using method overloading and various access modifiers.

this – How to use this keyword?

- This is a reference variable that refers to the current object. In JAVA, it represents current class object.

- **Usage of this keyword**
  - It can be used to refer current class instance variable.
  - this() can be used to invoke current class constructor.
  - It can be used to invoke current class method (implicitly)
  - It can be passed as an argument in the method call.
  - It can be passed as argument in the constructor call.
  - It can also be used to return the current class instance.

this keyword in java

- **Why use this keyword in JAVA?**
  - Main purpose: To differentiate the formal parameter and data members of class
  - Why?
    - whenever the formal parameter and data members of the class are similar then jvm gets ambiguity (no clarity between formal parameter and member of the class)
    - So to differentiate between formal parameter and data member of the class, the data member of the class must be preceded by "this".
  - "**this**" keyword can be use in two ways.
    - **this . (this dot)**
    - **this() (this off)**

this keyword - Scope

- **What is the scope of this keyword in JAVA**
  - The scope of "this" keyword is **within the class**
  - The main purpose of using "this" keyword in real life application is to differentiate variable of class or formal parameters of methods or constructor while working with **SIMILAR OBJECTS**
- **When do we require this keyword?**
  - The instance variables and arguments are same then we need to use this keyword.

    (To avoid ambiguity of JVM)
  - Different member variables and arguments then we do not require this keyword.

this() keyword - Usage

- **this()**
  - can be used to call one constructor within the another constructor without creation of objects multiple time for the same class
  - The this() constructor call can be used to invoke the current class constructor (constructor chaining)
  - This approach is better if you have many constructors in the class and want to reuse that constructor

- **Syntax**

```
this();                          // call default constructor
this(value1,value2,.....) // call parameterized constructor
```


this() invoked in Class Constructor

```
class Circle {
    float x, y, r;
    String cid;

    Circle() {
        System.out.println("New Circle is created!!");
    }

    Circle(float x) {
        this();
        this.x = x;
    }
    Circle(String cid) {
        this();
        this.cid = cid;
    }
```

this() invoked in Class Constructor

```
public static void main(String[] args) {
        Circle c;

        c = new Circle();
        System.out.println("x = " + c.x + ", y = " + c.y + ", r = " + c.r);

        c = new Circle(12);
        System.out.println("x = " + c.x + ", y = " + c.y + ", r = " + c.r);

        c1 = new Circle("Circle 1");
        System.out.println("CIRCLE ID: " + c1.cid + ", x = " + c1.x
                        + ", y = " + c1.y + ", r = " + c1.r);
}
```

this() Constructor Call

- this() constructor call should be used to reuse the constructor in the constructor.
- It maintains the chain between the constructors:

```
Class Person {
        int id;
        String name, email;
        Person(int id, String name) {
                this.id = id;
                this.name = name;
        }
        Person(int id, String name, String email) {
                this(id, name);
                this.email = email;
        }
}
```

this – invoke a Method

- **Example:**

```
Class Person {
        void show() {}
        void salary() {
                show();
        }
        public static void main(String
args[]) {
                Person p = new Person();
                p.salary();
        }
}
```

```
Class Person {
        void show() {}
        void salary() {
                this.show();
        }
        public static void main(String
args[]) {
                Person p = new Person();
                p.salary();
        }
}
```

- Here **this.** is added by Compiler

this – passing as an argument

- **Example:**

```
Class Person {
        void show(Person p1) {
        System.out.println("Showing P1 details!!");
        }
        void salary() {
                show(this);
        }
        public static void main(String args[]) {
                Person p = new Person();
                p.salary();
        }
}
```

Output:
Showing P1 details!!

this – invoke a Method

✧ **Example:**

```
Class P {
        Person obj;
        P (Person obj) {
                this.obj = obj;
        }
        void display() {
                System.out.println("Name: "
                        + obj.name);
        }
}
```

```
Class Person {
        String name = "Alfred De Souza";
        Person() {
                P p = new P(this);
                p.display();
        }
        public static void main(String
args[]) {
                Person p = new Person();
        }
}
```

Output: **Alfred De Souza**

this – return object of a class

```
Class Square {                          // Default Constructor is defined automatically

    int length, breadth;
    Square(int l, int b) {
        length = l;
        breadth = b;
    }
    Square getObject() {
        return this;
    }
    public static void main(String args[]) {
        Square s1 = new Square();
        Square s2;
        s2 = s1.getObject();
    }
}
```

You can print the detail of the object - Square

static – keyword in JAVA

- **static keyword**
  - The static keyword is used in java mainly for memory management
  - It is a keyword that are used for share the same variable or method of a given class
  - This is used for a constant variable or a method that is the same for every instance of a class. The main method of a class is generally labeled static
  - No object needs to be created to use static variable or call static methods, just put the class name before the static variable or method to use them
  - Static method can not call non-static method

- **Static Keyword can be used for Following**
  - Variable (also Known as Class Variable)
  - method (also Known as Class Method)
  - Block
  - Nested Class

static – Use of this keyword

- **static Variable**
  - If we declare any variable as **static**, it is known static variable.
  - The static variable can be used to refer the **common property of all objects** (that is not unique for each object).
  - Example:
    - company name – Common for all employees
    - college name of students - common for all students.
  - The static variable allocate memory only once in class area at the time of class loading.
  - Advantage of Static Variable
  - It makes your program memory efficient

static – When and Why?

- **When and Why we Use Static Variable?**
  - Suppose that we wish to store record of all employee of any company
    - Employee ID is unique for every employee
    - But company name is **common for all**
    - When we create a static variable as a company name then only once memory is allocated
    - Otherwise it allocate a memory space each time for every employee.

static – For Common Properties

- **Example**
  - Suppose there are 500 employees in a company
  - All instance data members will get memory each time when object is created
  - All Employees have their unique ID and NAME

  - Company - the common property of all objects
  - Make it static → memory is allotted only once

```
class Person {
        int empID;
        String name;
        static String company = "DCT";
}
```

Note: Java static Property is Shared to all Objects.

static – keyword in JAVA

- **Applicable to**
  - Method
  - Variable
  - Class Nested Within Another Class
  - Initialization Block
- **Not Applicable to**
  - Class (Not Nested)
  - Constructor
  - Interfaces
  - Method Local Inner Class(Difference then Nested Class)
  - Inner Class Methods
  - Instance Variables
  - Local Variables

static – Rules

- **static keyword rules**
  - Variable or Methods marked static belong to the Class rather then to any particular Instance
  - Static Method or Variable can be used without creating or referencing any instance of the Class
  - If there are instances, a static variable of a Class will be shared by all instances of that class, This will result in only one copy
  - A static Method can't access a non static variable nor can directly invoke non static Method (It can invoke or access Method or variable via instances)

static – Example

- **Let us look at the following:**

```
Class Person {
    int empID;
    String name;
    static String company = "DCT";

    Person(int id, String fname) {
        empID = id;
        name = fname;
    }

    void printInfo() {
        System.out.println("ID: " + id
            + ", Name: " + name
            + ", Company: " + company);
    }
}
```

```
public static void main(String args[]) {
    Person p1, p2;
    p1 = new Person(21, "John");
    p2 = new Person(17, "Peter");
}
```

Static vs non-static variables

- **Differences between static and non-static variables**

static variable	non-static variable
Known as class variables	Known as Instance variables
Memory is allotted whenever a class is created	Memory is allotted whenever an object or an instance of a class is created
Common for every object – memory location can be shared by every object	Specific to an object
Scope of these variables are global	Scope of these variables are local
Can be accessed with class reference	Can be accessed by with object reference
Syntax: **Classname.variablename**	**Syntax:** **Objectreference.variablename**

static – Create a counter

```
public class Counter {
    int count = 0;

    public Counter() {
        count++;
        System.out.println("Counter: " + count);
    }

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

static – Create a counter

```
public class Counter {
    static int count = 0;

    public Counter() {
        count++;
        System.out.println("Counter: " + count);
    }

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

Exercise - 5

- **Apply the Concepts for**
  - Trying various Algorithmic Techniques
  - Solving Complex Problems
- **Apply the following in every problem you attempt to solve:**
  - Object Oriented Design
  - Object Oriented Analysis
  - Object Oriented Programming
  - Object Oriented Thinking

Assignments / Penalties

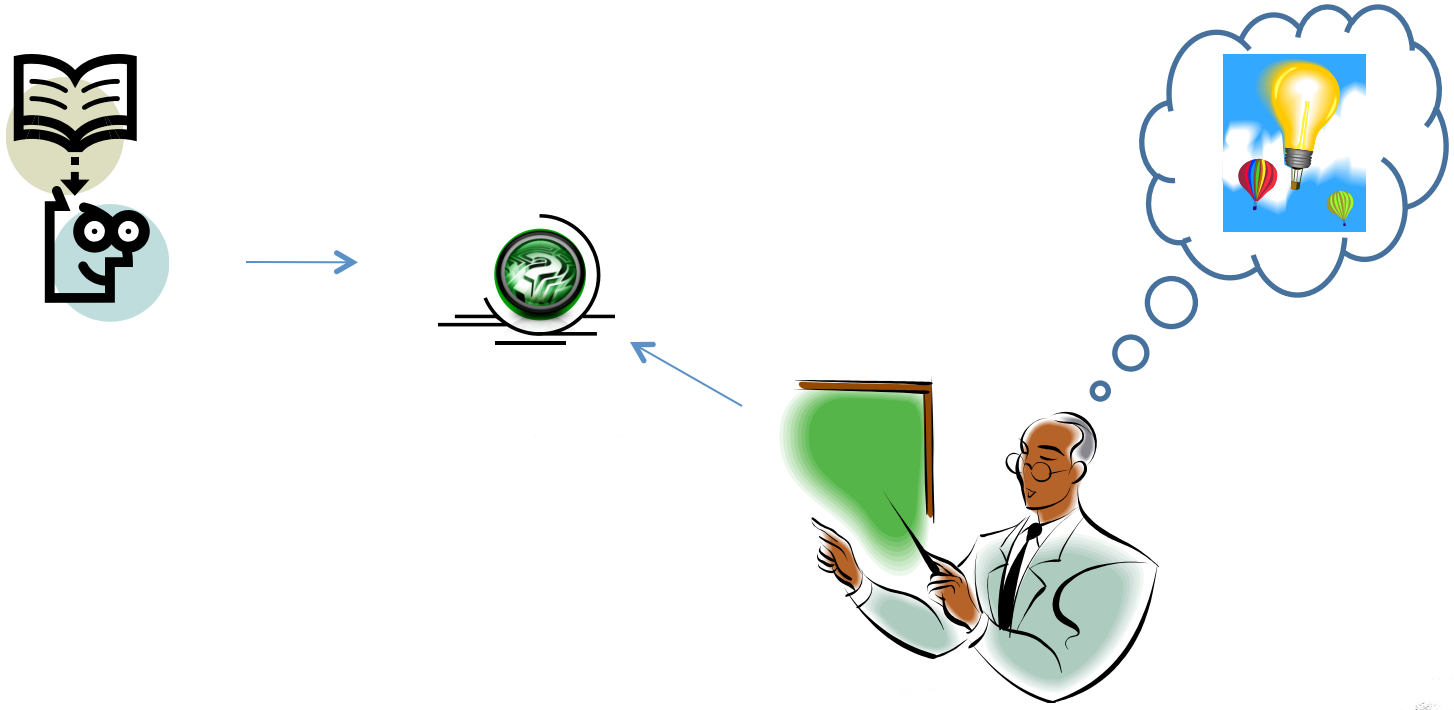
- Every Student is expected to complete the assignments and strictly follow a fair Academic Code of Conduct to avoid severe penalties

- Penalties would be heavy for those who involve in:
  - **Copy and Pasting** the code
  - **Plagiarism** (copied from your neighbor or friend – in this case, both will get "0" marks for that specific take home assignments)
  - If the candidate is **unable to explain his own solution**, it would be considered as a "copied case" !!
  - **Any other unfair means** of completing the assignments

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may leave me an email any time
  (email is the best way to reach me faster)


Thanks …



… Questions ???