JAVA TRAINING CENTER

Java Training Center

(No.1 in Training & placement)

OOPS WorkBook

Master the Content…

W.B-2

Author

Som Prakash Rai

JAVA TRAINING CENTER

Topic

Jtc 1: Example using Method Parameter

1) Jtc1.java

```
class Jtc1{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
MethodService serv = new MethodService();
serv.show('A');
byte by1 = 123;
serv.show(123);
serv.show(by1);
// serv.show(123L);
serv.show((int) 123L);
// serv.display(123);
serv.display((byte) 123);
serv.display(by1);
}
}
class MethodService {
void show(int ab) {
System.out.println("- show(int) \t:" + ab);
}
void display(byte by1) {
System.out.println("- display(byte) \t:" + by1);
}
}
```

Jtc 2: Example using Method Invocation

1) Jtc2.java

```
class Jtc2{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
```

```
int ab = 98;
System.out.println("ab in Main Before\t:" + ab);
Mno ref = new Mno();
ref.showValue(ab);
System.out.println("ab in Main After\t:" + ab);
}
}

class Mno {
void showValue(int ab) {
System.out.println("Before showValue\t:" + ab);
if (ab != 0)
showValue(ab / 10);
System.out.println("After showValue \t:" + ab);
}
}
```

Jtc 3: Example using Invoking Overloaded Methods

1) Jtc3.java

```
class Jtc3{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
byte by1 = 123;
OverloadManager mngr = new OverloadManager();
mngr.show(12, by1);
mngr.show(by1, 123);
// mngr.show(by1, by1);
mngr.show((int) by1, by1);
mngr.show(by1, (int) by1);
String str = null;
Object obj = null;
int arr[] = null;
mngr.display(str);
mngr.display(obj);
mngr.display(arr);
mngr.display(null);
mngr.showValues(arr);
```

```
mngr.showValues(str);
// mngr.showValues(null);
mngr.showValues((String) null);
mngr.showValues((int[]) null);
}
}

class OverloadManager {
void show(int ab, byte b1) {
System.out.println("** show(int,byte) **");
}
void show(byte b1, int ab) {
System.out.println("** show(byte,int) **");
}
void display(String str) {
System.out.println("__ display(String) __");
}
void display(Object obj) {
System.out.println("__ display(Object) __");
}
void showValues(String str) {
System.out.println("-- showValues(String) --");
}

void showValues(int[] arr) {
System.out.println("-- showValues(int[]) --");
}
}
```

Jtc 4: Example using Call By Value / Reference

1) Jtc4.java

```
class Jtc4{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
MethodParamService ref = new MethodParamService();

System.out.println("==== PRIMITIVE ====");
```

```
int ab = 123;
System.out.println("In main before\t:" + ab);
ref.modify(ab);
System.out.println("In main after\t:" + ab);

System.out.println("\n==== REFERENCE ====");
User ur = new User();
ur.uid = 101;
ur.phone = 6526668L;
System.out.println("In Main Before\t:" + ur.uid + "\t" + ur.phone);
ref.modify(ur);
System.out.println("In Main After\t:" + ur.uid + "\t" + ur.phone);
System.out.println("\n");
System.out.println("In Mai Before\t:" + ur.uid + "\t" + ur.phone);
ref.change(ur);
System.out.println("In Main After\t:" + ur.uid + "\t" + ur.phone);
}
}

class MethodParamService {
void modify(int ab) {
System.out.println("-- modify(int) ---");
System.out.println("Before Modifying\t:" + ab);
ab = ab + 1000;
System.out.println("After Modifying\t:" + ab);
}
void modify(User user) {
System.out.println("-- modify(User) ---");
System.out.println("Before Modifying\t:" + user.uid + "\t" + user.phone);
user = new User();
System.out.println("Before Modifying\t:" + user.uid + "\t" + user.phone);
user.uid = user.uid + 1000;
user.phone = 9999999999999L;
System.out.println("After Modifying\t:" + user.uid + "\t" + user.phone);
}

void change(User user) {
System.out.println("-- change(User) ---");
System.out.println("Before Modifying\t:" + user.uid + "\t" + user.phone);
user.uid = user.uid + 1000;
user.phone = 99999999L;
System.out.println("After Modifying data\t:" + user.uid + "\t"+ user.phone);
```

```
user = new User();
user.uid = 33333;
user.phone = 8583828785L;
System.out.println("After Modifying Ref\t:" + user.uid + "\t"+ user.phone);
}
}
class User {
int uid;
long phone;
}
```

Jtc 5: Example using Constructor Chaining

1) Jtc5.java

```
class Jtc5{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
new Employee(99).show();
System.out.println();
new Employee("Chandan").show();
System.out.println();
new Employee(98, "SomPrakash").show();
System.out.println();
new Employee(45, "Vikas", 6526668).show();
System.out.println();
new Employee(58, "Manish", 7676763290L, 85000.0F).show();
System.out.println();
new Employee(6526668, "Rai", 78562.00F).show();
}
}


class Employee {
int eid;
String name;
long phone;
float salary;
```

```
Employee(int eid) {
System.out.println("-- Employee(int) --\t:" + this);
this.eid = eid;
}
Employee(String name) {
// super(); //By Default
System.out.println("-- Employee(String) --\t:" + this);
this.name = name;
}
Employee(int eid, String name) {
this(eid);
// super();
System.out.println("-- Employee(int,String) --\t:" + this);
// Employee(eid);
// this(eid);
this.name = name;
}
Employee(int eid, String name, long phone) {
this(eid, name);
System.out.println("-- Employee(int,String,long) --\t:" + this);
this.phone = phone;
}
Employee(int eid, String name, long phone, float salary) {
this(eid, name, phone);
System.out.println("-- Employee(int,String,long,float) --\t:" + this);
this.salary = salary;
}

Employee(long phone, String name, float salary) {
this(name);
System.out.println("-- Employee(long,String,float) --\t:" + this);
this.phone = phone;
this.salary = salary;
}
void show() {
System.out.println(eid + "\t" + name + "\t" + phone + "\t" + salary);
}
}
```

Jtc 6: Example using Class Loading & Static Blocks Execution

1) Jtc6.java

```
class Jtc6{
```

```
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
System.out.println("** MAIN METHOD **");
}
static {
System.out.println("-- Static Block of Jtc6 --");
}
}
```

Jtc 7: Example using Class Loading & Static Blocks Execution

1) Jtc7.java

```
class Jtc7{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
System.out.println("** MAIN METHOD **");
TestClasses cl = null;
System.out.println("-- Ref Created --\n");
cl = new TestClasses();
System.out.println(cl);
}
}
class TestClasses {
static {
System.out.println("-- Static of TestClasses --");
}
TestClasses() {
System.out.println("-- TestClasses() Cons --");
}
}
```

Jtc 8: Example using Class Loading & Static Blocks Execution

1) Jtc8.java

```
class Jtc8{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
System.out.println("** MAIN METHOD **");
System.out.println(Xyz.var);
System.out.println("\n-- Value Accessed --");
System.out.println(new Xyz());
}
}
class Xyz {
static int var = 123;
static {
System.out.println("-- Static of Xyz --");
}
}
```

Jtc 9: Example using Class Loading & Static Blocks Execution

1) Jtc9.java

```
class Jtc9{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
System.out.println("** MAIN METHOD **");
new Student();
System.out.println("-- Student Object Created --\n");
new Employee();
System.out.println("-- Employee Object Created --\n");
}
}
class Person {
static {
System.out.println("\n-- Static of Person --");
```

```
}
Person() {
System.out.println("-- Person() Cons --");
}
}

class Student extends Person {
static {
System.out.println("-- Static of Student --");
}
Student() {
System.out.println("-- Student() Cons --");
}
}
class Employee extends Person {
static {
System.out.println("-- Static of Employee --");
}
}
```

Jtc 10: Example using Class Loading & Static Blocks Execution

1) Jtc10.java

```
class Jtc10{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
System.out.println("** MAIN METHOD **");
System.out.println(Mno.VAL);// SOP(9090);
System.out.println(Mno.VAL+100);// SOP(9190);
}
}
class Mno {
final static int VAL = 9090;
static {
System.out.println("** STATIC BLOCK OF Mno\t:" + VAL);
}
}
```

Jtc 11: Example using Loading the Class Dynamically

1) Jtc11.java

```
class Jtc11{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args)throws Exception {
if (args.length == 1) {
String cName = args[0];
Class cl = Class.forName(cName);
Class cl1 = Class.forName(cName);
System.out.println(cl);
System.out.println(cl1);
System.out.println(cl==cl1);
System.out.println("-- Class Loaded \t:" + cl.getName());
} else {
System.out.println("Provide Classname as CLA");
}
}
}
//TestClasses , Person  , Student and Employee class same as Previous Examples
```

Jtc 12: Example using Loading the Class Dynamically

1) Jtc12.java

```
class Jtc12{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) throws Exception {
if (args.length == 1) {
String cName = args[0];
ClassLoader load = Test60.class.getClassLoader();
Class cl = Class.forName(cName, false, load);
System.out.println("-- Class Loaded \t:" + cl.getName());
```

```
System.out.println("SuperClass\t:" + cl.getSuperclass());
System.out.println("\n================");
Object obj = cl.newInstance();
System.out.println(obj);
} else {
System.out.println("Provide Classname as CLA");
}
}
}
//TestClasses , Person  , Student and Employee class same as Previous Examples
```

Jtc 13: Example using Loading the Class Dynamically

1) Jtc13.java

```
class Jtc13{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) throws Exception {
if (args.length == 1) {
String cName = args[0];
ClassLoader load = Test61.class.getClassLoader();
Class cl = load.loadClass(cName);
System.out.println("-- Class Loaded \t:" + cl.getName());
System.out.println("SuperClass\t:" + cl.getSuperclass());
System.out.println("\n================");
Object obj = cl.newInstance();
System.out.println(obj);
} else {
System.out.println("Provide Classname as CLA");
}
}
}
//TestClasses , Person  , Student and Employee class same as Previous Examples
```

Jtc 14: Example to show that class will be loaded only once

1) Jtc14.java

```
class Jtc14{
/*
```

```
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) throws Exception {
System.out.println("In Main Method Loading the Hello Class");
ClassLoader loader = Test62.class.getClassLoader();
Class.forName("Hello", false, loader);
System.out.println("-- Class Loaded Successfully --");
System.out.println("Delete the .class file and press ENTER ");
System.in.read();
Hello h = new Hello();
h.show();
h.display();
new Hello(12).show();
new Hello(89, "JTC").show();
new Hello().display();
System.out.println(Hello.value);
}
}
class Hello {
int ab;
String msg;
static int value = 1234;
Hello() {
System.out.println("-- Hello() Cons --");
}
Hello(int ab) {
System.out.println("-- Hello(int) Cons --");
}
Hello(int ab, String msg) {
System.out.println("-- Hello(int,String) Cons --");
}
void show() {
System.out.println("** show () in Hello **");
System.out.println(ab);
System.out.println(msg);
}
void display() {
System.out.println("** display() in Hello **");
}
```

```
}
```

Jtc 15: Example using Static Inner Class

1) Jtc15.java

```
class Jtc15{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
System.out.println(MyOuterClass.MyInnerClass.LENGTH);
System.out.println(MyOuterClass.MyInnerClass.val);
MyOuterClass.MyInnerClass.displayMessage();
//MyInnerClass ref1 = null;
MyOuterClass.MyInnerClass ref = null;
System.out.println(ref.msg);
ref = new MyOuterClass.MyInnerClass("Message from Main Method");

//ref=new MyOuterClass().new MyInnerClass("Message from Main Method");
//ref=new MyOuterClass().MyInnerClass("Message from Main Method");
System.out.println(ref.msg);
MyOuterClass.displayInOuterClass();
new MyOuterClass().showInOuterClass();
ref.showResult();
}
}


class MyOuterClass {
int result = 1045;
static int value = 1212;

static class MyInnerClass {
static int val = 9090;
static final int LENGTH = 10;
String msg;
MyInnerClass(String msg) {
this.msg = msg;
}
static void displayMessage() {
```

```
System.out.println("-- displayMessage static in Inner Class --");
}
void showResult() {
System.out.println("\n-- showResult () in Inner Class--");
System.out.println("MSG\t:" + msg);
System.out.println("val\t:" + val);
System.out.println("** Member of Outer Class **");
System.out.println("static\t:" + value);
// System.out.println("ins\t:"+result);
System.out.println("ins\t:" + new MyOuterClass().result);
}
}

void showInOuterClass() {
System.out.println("\n-- showInOuterClass --");
MyInnerClass ref1 = new MyInnerClass("Message in Show Method");
System.out.println(ref1.msg);
}
static void displayInOuterClass() {
System.out.println("\n-- displayInOuterClass --");
MyInnerClass ref1 = new MyInnerClass("Message in Display Method");
System.out.println(ref1.msg);
}
}
```

Jtc 16: Example using Instance Inner Class

1) Jtc16.java

```
class Jtc16{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
A$B ref = null;
ref = new A$B();
System.out.println(ref);
// X$Y ref2=null;
X.Y ref3 = null;

X ref4 = new X();
```

```
// ref3=new Y();
ref3 = ref4.new Y();
ref3 = new X().new Y();
System.out.println(ref3);
System.out.println("\n\n--------------");
Hello.display();
System.out.println();
Hello he = new Hello("MSG in MAIN");
System.out.println(he.msg);
// System.out.println(he.intValue);
he.show();
Hello.JTCInner1 inRef = null;
inRef = new Hello("MSG IN MAIN AGAIN").new JTCInner1(6060);
inRef.showDataInInnerClass();
inRef = he.new JTCInner1(4040);
inRef.showDataInInnerClass();
System.out.println(inRef.intValue);
// System.out.println(inRef.msg);
}
}

class A$B { }
class B { }
class A {
// class B{}
}

class X {
int var = 10;
class Y { }
}

class Y {}

// class X$Y{}

class Hello {
static int VAL = 9090;
String msg;

Hello(String msg) {
this.msg = msg;
```

```
}

class JTCInner1 {
// static int stValue=9876;
final static int CONS = 9999;
int intValue;
public JTCInner1(int intValue) {
this.intValue = intValue;
}
void showDataInInnerClass() {
System.out.println("\n** showData in Inner Class **");
System.out.println(intValue);
System.out.println(CONS);
System.out.println(msg);
// System.out.println(this.msg);
System.out.println(VAL);
}
}

void show() {
System.out.println("\n-- INSTANCE Show Method ");
System.out.println(VAL);
System.out.println(msg);
System.out.println(this.msg);
JTCInner1 ref = null;
ref = new JTCInner1(1111);
ref = this.new JTCInner1(1111);
System.out.println(ref.intValue);
ref.showDataInInnerClass();
}

static void display() {
System.out.println("\n--- Static Display Method --");
System.out.println(VAL);
// System.out.println(msg);
Hello ref = new Hello("MSG in Display");
System.out.println(ref.msg);
JTCInner1 ref1 = null;
// ref1=new JTCInner1(2222);
ref1 = ref.new JTCInner1(2222);
ref1.showDataInInnerClass();
}
```

```
}
```

Jtc 17: Example using Referencing the Object from Inner Class

1) Jtc17.java

```
class Jtc17{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
Abc ref=new Abc("WELCOME");
Abc.Pqr ref2=ref.new Pqr(3232);
ref2.show();
System.out.println();
new Abc("THANKS").new Pqr(4141).show();
}
}
class Abc {
static String stVar = "STATIC IN Outer";
String var;
Abc(String var) {
this.var = var;
}
class Pqr {
final static String stVar = "STATIC IN Inner";
int var;
Pqr(int var) {
this.var = var;
}
void show() {
boolean var = false;
System.out.println("\n-- in Show Method -- in Inner Class --");
System.out.println(var);
System.out.println(this);
System.out.println(this.var);
System.out.println(Abc.stVar);
System.out.println(Pqr.stVar);
System.out.println(Abc.this);
System.out.println(Abc.this.var);
```

```
}
}
}
```

Jtc 18: Example using Abstract Class

1) Jtc18.java

```
class Jtc18{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
System.out.println(Person.minAgeToVote);
Person per = null;
// per=new Person();
// per.showWorkInfo();
// System.out.println(per.name);

per = new Employee("SomPrakash", 7676763290L);
per.showWorkInfo();
System.out.println(per.name + "\t" + per.phone);

per = new OldStudent();
per.showWorkInfo();
per = new CurrentStudent();
per.showWorkInfo();
System.out.println("\n**************");
Person p = null;
p = PersonService.getInstance("Employee");
System.out.println(p.getPersonType());
p = PersonService.getInstance("OldStudent");
System.out.println(p.getPersonType());
p = PersonService.getInstance("CurrentStudent");
System.out.println(p.getPersonType());
}
}

abstract class Person {
static int minAgeToVote = 18;
String name;
```

```
long phone;

Person(String name, long phone) {
this.name = name;
this.phone = phone;
System.out.println("-- Person(String,long) Cons.. \t:" + this);
}
Person() {
System.out.println("-- Person() Cons.. \t:" + this);
}
{
System.out.println("\n** Person Instance Block **");
}

abstract void showWorkInfo();
abstract String getPersonType();
void show() {
// Person p=new Person();
// p.showWorkInfo();
}
}
// class Farmer extends Person{ }

class Employee extends Person {
Employee(String name, long phone) {
super(name, phone);
}
Employee() { }
String getPersonType() {
return "Employee";
}
void showWorkInfo() {
System.out.println("-- Employee Working in Company --");
}
}

abstract class Student extends Person { }

class CurrentStudent extends Student {
void showWorkInfo() {
System.out.println("-- Attending the classes --");
}
```

```
String getPersonType() {
return "CurrentStudent";
}
}

class OldStudent extends Student {
void showWorkInfo() {
System.out.println("-- Searching for Job --");
}
String getPersonType() {
return "OldStudent";
}
}

class PersonService {
static Person getInstance(String cName) {
if (cName.equals("Employee"))
return new Employee();
else if (cName.equals("OldStudent"))
return new OldStudent();
else if (cName.equals("CurrentStudent"))
return new CurrentStudent();
else
return null;
}
}
```

Jtc 19: Example using Limitation of Multiple Inheritance

1) Jtc19.java

```
class Jtc19{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
interface Inter11 {
void show();
}
interface Inter22 {
int show();
}
```

```
class Cd implements Inter11,Inter22 {
/*
public void show() {}
*/
public int show() {}
}

class Ab implements Inter11 {
public void show() { }
}

class Bc implements Inter22 {
public int show() {
return 0;
}
}
```

Jtc 20: Example using Dynamic Dispatch and Dynamic Polymorphism

Shape.java

```
public abstract class Shape {
double length;
Shape(double length) {
this.length = length;
}
final double getLength() {
return this.length;
}
abstract double findArea();
abstract String getType();
static void display() {
System.out.println("-- Static Display in Shape --");
}
}
```

ShapeUtil.java

```
public class ShapeUtil {
public void showShapeInfo(Shape sp) {
System.out.println("\nType\t:" + sp.getType());
System.out.println("Length\t:" + sp.getLength());
```

```
System.out.println("Area\t:" + sp.findArea());
sp.display();// Shape.display();
// sp.getSide();
// sp.getWidth();
System.out.println();
if (sp instanceof Square) {
Square sq = (Square) sp;
System.out.println("Side\t:" + sq.getSide());
sq.display();
} else if (sp instanceof Rectangle) {
Rectangle rec = (Rectangle) sp;
rec.display();
System.out.println("Width\t:" + rec.getWidth());
}
}
}
```

Square.java

```
public class Square extends Shape {
Square(double side) {
super(side);
}
double findArea() {
System.out.println("** Square Find Area **");
return length * length;
}
double getSide() {
return length;
}
String getType() {
return "Square";
}
static void display() {
System.out.println("== Static Display in Square ==");
}
}
```

Rectangle.java

```
public class Rectangle extends Shape {
double width;
```

```
Rectangle(double length, double width) {
super(length);
this.width = width;
}
double findArea() {
System.out.println("** Rectangle Find Area **");
return length * width;
}
String getType() {
return "Rectangle";
}
double getWidth() {
return width;
}

static void display() {
System.out.println("++ Static Display in Rectangle ++");
}
}
```

1) Jtc20.java

```
class Jtc20{
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public static void main(String[] args) {
ShapeUtil util = new ShapeUtil();
util.showShapeInfo(new Square(12.0));
util.showShapeInfo(new Rectangle(12.0, 10.0));
}
}
```