JAVA TRAINING CENTER

Java Training Center

(No.1 in Training & placement)

Reflection WorkBook

Master the Content…

W.B-4

Author

Som Prakash Rai

JAVA TRAINING CENTER

Topic

Jtc 1: Example using Reflection

Jtc 2: Example using Reflection.

Jtc 3: Example using Reflection.

Jtc 4: Accessing the Class Member information Dynamically

Jtc 5: Accessing the method information Dynamically.

Jtc 6: Invoking the Method Dynamically

Jtc 7: Creating the Object Dynamically using Various Constructor

Jtc 1: Reflection Example

```
package com.jtc;
import java.lang.reflect.*;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public class Jtc1 {
public static void main(String[] args) {
try{
Hello hh=new Hello();
hh.show();
Class cls=Class.forName("com.jtc.Hello");
Hello h=(Hello)cls.newInstance();
h.show();
System.out.println(cls);
System.out.println(cls.getName());
System.out.println(cls.getPackage());
System.out.println(cls.getPackage().getName());
System.out.println(cls.getSuperclass());
System.out.println(cls.getSuperclass().getName());
Class c[]=cls.getInterfaces();
for(int i=0;i<c.length;i++){
System.out.println(c[i].getName());
}
System.out.println(cls.getModifiers());
System.out.println(Modifier.PUBLIC);
System.out.println(Modifier.FINAL);
}catch(Exception e){
e.printStackTrace();
}
}
}

interface i1{}
interface i2{}
class A{}
final class Hello extends A implements i1,i2{
```

```
int a=10;
int b;
Hello(){ }
Hello(int b){
this.b=b;
}
void show(){
System.out.println(a);
System.out.println(b);
}
}
```

Jtc 2: Reflection Example

```
package com.jtc;
import java.lang.reflect.*;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public class Jtc2 {
public static void main(String[] args) {
try{
Class cls=Class.forName("com.jtc.Hello");
Hello h=(Hello)cls.newInstance();
h.show();
System.out.println(Modifier.PUBLIC);
System.out.println(Modifier.FINAL);
System.out.println(Modifier.STATIC);

// fields
System.out.println("fields");
Field f[]=cls.getFields();
for(int i=0;i<f.length;i++){
System.out.println(f[i].getModifiers()+"..."+f[i].getType()+"..."+f[i].getName());
}


System.out.println("Declared fields");
// Declared fields
```

```
Field ff[]=cls.getDeclaredFields();
for(int i=0;i<ff.length;i++){
System.out.println(ff[i].getModifiers()+"..."+ff[i].getType()+"..."+ff[i].getName());
}

// methods
System.out.println("methods");
Method m[]=cls.getMethods();
for(int i=0;i<m.length;i++){
System.out.println(m[i].getModifiers()+"..."+m[i].getReturnType()+"..."+m[i].getName()
);
}
// Declared methods
System.out.println("Declared methods");
Method mm[]=cls.getDeclaredMethods();
for(int i=0;i<mm.length;i++){
System.out.println(mm[i].getModifiers()+"..."+mm[i].getReturnType()+"..."+mm[i].get
Name());
}
// constructors
System.out.println("constructors");
Constructor c[]=cls.getConstructors();
for(int i=0;i<c.length;i++){
System.out.println(c[i].getModifiers()+"..."+c[i].getName());
}
//invoking methods dynamically
mm=cls.getDeclaredMethods();
for(int i=0;i<mm.length;i++){
if(mm[i].getName().equals("show"))
mm[i].invoke(h,null);
}
}catch(Exception e){
e.printStackTrace();
}
}
}
class Hai {
public int x=200;
public void hai(){
System.out.println("hai");
}
}
```

```
class Hello extends Hai{
public static final int a=10;
public static int b=20;

public Hello(){ }

public void show(){
System.out.println("show");
System.out.println(a);
System.out.println(b);
}
public void m1(String s1,String s2){
System.out.println("m1");
System.out.println(s1);
System.out.println(s2);
}
public void m2(int x,int y){
System.out.println("m2");
System.out.println(x);
System.out.println(y);
}
}
```

Jtc 3: Reflection Example

```
package com.jtc;

import java.lang.reflect.*;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public class Jtc3 {
public static void main(String[] args) {
try{
Class cls=Class.forName("com.jtc.Hello");
Hello h=(Hello)cls.newInstance();
//invoking methods dynamically
Method []mm=cls.getDeclaredMethods();
```

```
for(int i=0;i<mm.length;i++){
if(mm[i].getName().equals("show")){
mm[i].invoke(h,null);
}
if(mm[i].getName().equals("m1")){
Object o[]={"jtc1","jtc2"};
mm[i].invoke(h,o);
}
if(mm[i].getName().equals("m2")){
Object o[]={new Integer(99),new Integer(88)};
mm[i].invoke(h,o);
}
}
}catch(Exception e){
e.printStackTrace();
}
}
}
class Hello {
public void show(){
System.out.println("show");
}
public void m1(String s1,String s2){
System.out.println("m1");
System.out.println(s1);
System.out.println(s2);
}
public void m2(int x,int y){
System.out.println("m2");
System.out.println(x);
System.out.println(y);
}
}
```

Jtc 4: Reflection Example

```
import java.lang.reflect.*;
import java.io.*;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
```

```
*@Call        :+91-9990399111
* */
public class Jtc4{
public static void main(String str[]) throws Exception{
Class cls=Class.forName("Student");
System.out.println("*** INTERFACES  ***");
Class inters[]=cls.getInterfaces();
for(int i=0;i<inters.length;i++){
System.out.println(inters[i]);
}
System.out.println("*** PUBLIC CONSTRUCTOR ***");
Constructor pubCons[]=cls.getConstructors();
for(int i=0;i<pubCons.length;i++){
System.out.println(pubCons[i]);
}
System.out.println("*** Declared CONSTRUCTOR ***");
Constructor decCons[]=cls.getDeclaredConstructors();
for(int i=0;i<decCons.length;i++){
System.out.println(decCons[i]);
}
System.out.println("*** PUBLIC FIELD ***");
Field pubFlds[]=cls.getFields();
for(int i=0;i<pubFlds.length;i++){
System.out.println(pubFlds[i]);
}
System.out.println("*** DECLARED FIELD ***");
Field decFlds[]=cls.getDeclaredFields();
for(int i=0;i<decFlds.length;i++){
System.out.println(decFlds[i]);
}
System.out.println("*** PUBLIC INNER CLASSES ***");
Class innClass[]=cls.getClasses();
for(int i=0;i<innClass.length;i++){
System.out.println(innClass[i]);
}
}
}
interface Inter1{}
interface Inter2{}
interface Inter3{}
class User{
public String username;
```

```
String password;
public String name;
public class Ab{}
class Bc{}
}
class Student extends User implements Inter1,Inter2,Inter3,Cloneable,Serializable{
private String batchId;
public String batchTimings;
int sid;
Student(int ab){}
private Student(int ab,String nm){}
public Student(int ab,long phone){}
protected Student(int ab,boolean registered){}
public Student(int ab,String nm,long phone){}
protected Student(String nm,float fee){}
Student(String nm){}
class A{}
public class B{}
}
```

Jtc 5: Accessing the Method Information

```
import java.lang.reflect.*;
import java.io.*;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call       :+91-9990399111
* */
public class Jtc5{
public static void main(String str[]) throws Exception{
Class cls=Class.forName("Student");
Object st=new Student();
Object test=new RefTest5();
System.out.println(cls.isInstance(st));
System.out.println(cls.isInstance(test));

Method methods[]=cls.getDeclaredMethods();
for(int i=0;i<methods.length;i++){
Method m=methods[i];
```

```
System.out.println("\n=======================");
System.out.println(m);
System.out.println("**NAME***\t:"+m.getName());
Class retType=m.getReturnType();
System.out.println("**RET TYPE***\t:"+retType.getName());
System.out.println("Interface\t:"+retType.isInterface());
System.out.println("Primitive\t:"+retType.isPrimitive());
System.out.println("Arrays\t:"+retType.isArray());
Class params[]=m.getParameterTypes();
for(int j=0;j<params.length;j++){
System.out.print(params[j]+" , ");
}
int x=m.getModifiers();
System.out.println("\nPUBLIC\t:"+Modifier.isPublic(x));
System.out.println("FINAL\t:"+Modifier.isFinal(x));
System.out.println("STATIC\t:"+Modifier.isStatic(x));
System.out.println(Modifier.toString(x));
}
}
}
class Student{
public synchronized void show(int ab,String name,boolean valid,long phone ){
}
static final public String getName(int id,String email){
return "";
}
public int showInformation(String email){
return 0;
}
int[] getStudentIds(){
return null;
}
}
```

Jtc 6: Invoking the Method Dynamically

```
package com.jtc.ref.obj;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
```

```
*@Call        :+91-9990399111
* */

import java.io.*;
import java.lang.reflect.*;

public class Jtc6 {
public static void main(String[] args) throws Exception {
if (args.length == 1) {
String cName = args[0];
Class cl1 = Class.forName(cName);
DataInputStream dis = new DataInputStream(System.in);
Method ms[] = cl1.getDeclaredMethods();
for (int i = 0; i < ms.length; i++) {
Method m = ms[i];
System.out.println("\nCalling the method " + m.getName());
Object objReq = null;
int mod = m.getModifiers();
boolean st = Modifier.isStatic(mod);
if (!st) {
objReq = cl1.newInstance();
}
Class params[] = m.getParameterTypes();
Object[] reqArgs = new Object[params.length];
for (int j = 0; j < params.length; j++) {
String type = params[j].getName();
System.out.println("Enter Value of type\t:" + type);
String val = dis.readLine();
if (type.equals("boolean"))
reqArgs[j] = new Boolean(val);
else if (type.equals("byte"))
reqArgs[j] = new Byte(val);
else if (type.equals("char"))
reqArgs[j] = new Character(val.charAt(0));
else if (type.equals("short"))
reqArgs[j] = new Short(val);
else if (type.equals("int"))
reqArgs[j] = new Integer(val);
else if (type.equals("long"))
reqArgs[j] = new Long(val);
else if (type.equals("float"))
reqArgs[j] = new Float(val);
```

```
else if (type.equals("double"))
reqArgs[j] = new Double(val);
else if (type.equals("java.lang.String"))
reqArgs[j] = val;
}
Object res = m.invoke(objReq, reqArgs);
System.out.println("Result is\t:" + res);
}
} else {
System.out.println("Provide fully qualified class name & Method Name as CLA");
}
}
}

class Student {
private void showDetails() {
System.out.println("-- showDetails () of Student\t:" + this);
}

static int searchId(int ab, String bc, boolean b1) {
System.out.println("-- searchId () of Student\t:");
System.out.println("ab\t:" + ab);
System.out.println("bc\t:" + bc);
System.out.println("b1\t:" + b1);
return 99;
}

boolean verifyStudent(char ch, String bc, boolean b1, long val, float f1) {
System.out.println("-- verifyStudent () of Student\t:" + this);
System.out.println("ch\t:" + ch);
System.out.println("bc\t:" + bc);
System.out.println("b1\t:" + b1);
System.out.println("val\t:" + val);
System.out.println("f1\t:" + f1);
return true;
}
}
```

Jtc 7: Creating the Object Dynamically using Various Constructor

```
package com.jtc.ref.obj;
```

```
import java.io.*;
import java.lang.reflect.*;

public class Jtc7 {
public static void main(String[] args) throws Exception {
if (args.length != 1) {
System.out.println("Provide the fully qualified class name as CLA");
System.exit(0);
}
DataInputStream dis = new DataInputStream(System.in);

String className = args[0];
Class cl = Class.forName(className);
Constructor cons[] = cl.getConstructors();
for (int k = 0; k < cons.length; k++) {
Constructor c = cons[k];

Class cParams[] = c.getParameterTypes();
Object objArg[] = new Object[cParams.length];
for (int i = 0; i < cParams.length; i++) {
Class p = cParams[i];
String type = p.getSimpleName();
System.out.println("Enter value of type\t:" + type);
String val = dis.readLine();
if (type.equals("boolean")) {
objArg[i] = new Boolean(val);
} else if (type.equals("byte")) {
objArg[i] = new Byte(val);
} else if (type.equals("short")) {
objArg[i] = new Short(val);
} else if (type.equals("char")) {
objArg[i] = new Character(val.charAt(0));
} else if (type.equals("int")) {
objArg[i] = new Integer(val);
} else if (type.equals("long")) {
objArg[i] = new Long(val);
} else if (type.equals("float")) {
objArg[i] = new Float(val);
} else if (type.equals("double")) {
objArg[i] = new Double(val);
} else if (type.equals("String")) {
```

```
objArg[i] = val;
} else {
objArg[i] = p.newInstance();
}
}
Object obj = c.newInstance(objArg);
System.out.println(obj);
}
}
}
```

2) Employee.java

```
package com.jtc.ref.obj;

public class Employee {
private int eid;
private String eName;
private String email;
private long phone;
private boolean permanent;
private Address empAdd;

public Employee(int eid, String email, long phone, Address empAdd) {
this.eid = eid;
this.email = email;
this.phone = phone;
this.empAdd = empAdd;
}

public Employee(int eid, String eName, String email, long phone,
boolean permanent, Address empAdd) {
this.eid = eid;
this.eName = eName;
this.email = email;
this.phone = phone;
this.permanent = permanent;
this.empAdd = empAdd;
}

public String toString() {
```

```
return "Employee [eid=" + eid + ", eName=" + eName + ", email=" + email
+ ", phone=" + phone + ", permanent=" + permanent + ", empAdd="
+ empAdd + "]";
}

}
```

3) Address.java

```
package com.jtc.ref.obj;

public class Address {
public String toString() {
return "Emp Address";
}
}
```

java.lang Package

```
public final class Class extends Object implements Serializable{
   public native int getModifiers();
   public native boolean isArray();
   public native boolean isInterface();
   public native boolean isPrimitive();
   public native Class getDeclaringClass();
   public native Class getSuperclass();
   public Class[] getClasses();
   public Class[] getDeclaredClasses();
   public native Class[] getInterfaces();
   public Object newInstance();
   public native Object[] getSigners();
   public native boolean isInstance(Object);
   public Package getPackage();
   public native String getName();
   public String toString();
   public Constructor[] getConstructors();
   public Constructor[] getDeclaredConstructors();
   public Field[] getDeclaredFields();
   public Field[] getFields();
   public Method[] getDeclaredMethods();
   public Method[] getMethods();
```

```
    public static Class forName(String);
    static native Class getPrimitiveClass(String);
    public Constructor getConstructor(Class[]);
    public Constructor getDeclaredConstructor(Class[]);
    public Field getDeclaredField(String);
    public Field getField(String);
    public Method getDeclaredMethod(String,Class[]);
    public Method getMethod(String,Class[]);
}
```

java.lang.reflect API

```
public final class Constructor implements Member{
    public int getModifiers();
    public Class getDeclaringClass();
    public Class[] getExceptionTypes();
    public Class[] getParameterTypes();
    public boolean equals(Object);
    public String getName();
    public String toString();
    public Object newInstance(Object[]);
    Constructor(Class,Class[],Class[],int,int);
}
public final class Field implements Member{
    public int getModifiers();
    public Class getDeclaringClass();
    public Class getType();
    public byte getByte(Object);
    public char getChar(Object);
    public double getDouble(Object);
    public float getFloat(Object);
    public int getInt(Object);
    public long getLong(Object);
    public short getShort(Object);
    public boolean equals(Object);
    public boolean getBoolean(Object);
    public void setByte(Object,byte);
    public void setChar(Object,char);
    public void setDouble(Object,double);
    public void setFloat(Object,float);
    public void setInt(Object,int);
    public void setLong(Object,long);
```

```
    public void setShort(Object,short);
    public void setBoolean(Object,boolean);
    public String getName();
    public String toString();
    public Object get(Object);
    public void set(Object,Object);
    static String getTypeName(Class);
    Field(Class,String,Class,int,int);
}
public final class Method extends AccessibleObject implements Member{
    public int getModifiers();
    public Class getDeclaringClass();
    public Class getReturnType();
    public Class[] getExceptionTypes();
    public Class[] getParameterTypes();
    public boolean equals(Object);
    public String getName();
    public String toString();
    public Object invoke(Object,Object[]);
    Method(Class,String,Class[],Class,Class[],int,int);
}

public class Modifier extends Object{
    public static final int PUBLIC;
    public static final int PRIVATE;
    public static final int PROTECTED;
    public static final int STATIC;
    public static final int FINAL;
    public static final int SYNCHRONIZED;
    public static final int VOLATILE;
    public static final int TRANSIENT;
    public static final int NATIVE;
    public static final int INTERFACE;
    public static final int ABSTRACT;
    public static final int STRICT;
    public static boolean isAbstract(int);
    public static boolean isFinal(int);
    public static boolean isInterface(int);
    public static boolean isNative(int);
    public static boolean isPrivate(int);
    public static boolean isProtected(int);
    public static boolean isPublic(int);
```

```
    public static boolean isStatic(int);
    public static boolean isStrict(int);
    public static boolean isSynchronized(int);
    public static boolean isTransient(int);
    public static boolean isVolatile(int);
}
```