**First Program -**

**public** **class** immutableString {

**public** **static** **void** main(String[] args) {

String str1 = **new** String("Abcd");

String str2 = **new** String("Abcd");

// checking if str1 and str2 points to the same object or not

System.***out***.println("Comparing references to objects created using new keyword:");

**if**(str1==str2){

System.***out***.println("str1 and str2 are reference to the same object");

}

**else**{

System.***out***.println("str1 and str2 are references of different objects in memory");

}

System.***out***.println();

// checking if str1 and str2 points to the same object or not

String s1 = "Hello";

String s2 = "Hello";

System.***out***.println("Comparing references to objects created using literals:");

**if**(s1==s2){

System.***out***.println("str1 and str2 are reference to the same object");

}

**else**{

System.***out***.println("str1 and str2 are references of different objects in memory");

}

System.***out***.println();

// Now printing the addresses of strings and confirming again

String a = "java";

System.***out***.println("initial hash code of a- " + a.hashCode());

a = "c++";

System.***out***.println("modified hash code of a- "+a.hashCode());

String b = "java";

System.***out***.println("hash code of b- "+b.hashCode());

}

}

**Second program -**

**final** **class** immutableUser {

**private** **final** String firstName;

**private** **final** String lastName;

**private** **final** String address;

// all fields are immutable

**public** immutableUser(String firstName, String lastName, String address) {

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.address = address;

}

**public** String getFirstName() {

**return** firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** String getAddress() {

**return** address;

}

}

**public** **class** immutableUserDemo{

**public** **static** **void** main(String[] args) {

immutableUser u = **new** immutableUser("abhishek", "jain", "gwalior");

System.***out***.println("Once created, object u is immutable as there is no setter functions, nor are the fields accessible directly as they are private");

}

}

**Third Program -**

**public** **class** mutableAddress {

**private** String city;

**private** Integer zipCode;

mutableAddress(String city, Integer zipCode){

**this**.city = city;

**this**.zipCode = zipCode;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String toString() {

**return** "Address [city=" + city + ", zipCode=" + zipCode + "]";

}

**public** Integer getZipCode() {

**return** zipCode;

}

**public** **void** setZipCode(Integer zipCode) {

**this**.zipCode = zipCode;

}

}

**Fourth Program -**

**final** **class** Student{

**final** **private** String name; // immutable field

**final** **private** mutableAddress address; // mutable field

**public** Student(String name, mutableAddress address) {

**this**.name = name;

**this**.address = address;

}

**public** String toString() {

**return** "Student [name=" + name + ", address=" + address + "]";

}

**public** String getName() {

**return** name;

}

**public** mutableAddress getAddress() {

**return** address;

}

}

**public** **class** mutableFieldsDemo {

**public** **static** **void** main(String[] args) {

mutableAddress address = **new** mutableAddress("gwalior", 474002);

System.***out***.println("address is "+address);

Student s = **new** Student("abhi jain", address);

System.***out***.println("\n object constructed by passing address in the constructor : \n" + s);

address.setCity("delhi");

System.***out***.println("\n Changing the local reference variable address to : "+address);

System.***out***.println("Object also gets modified: "+s);

System.***out***.println("\n Thus Class Student is not an immutable class as we were able to change content of an object of class Student");

}

}

**Fifth Program -**

**final** **class** immutableStudent{

**final** **private** String name; // immutable field

**final** **private** mutableAddress address; // mutable field

**public** immutableStudent(String name, mutableAddress address) {

**this**.name = name;

**this**.address = **new** mutableAddress(address.getCity(),address.getZipCode()); // performing deep copy in the constructor for mutable field

}

**public** String toString() {

**return** "Student [name=" + name + ", address=" + address + "]";

}

**public** String getName() {

**return** name;

}

**public** mutableAddress getAddress() {

**return** **new** mutableAddress(address.getCity(), address.getZipCode()); //not sending the reference to the original object, instead creating a copy

// return address;

}

}

**public** **class** immutableClassWithMutableFields {

**public** **static** **void** main(String[] args) {

mutableAddress address = **new** mutableAddress("gwalior", 474002);

immutableStudent s = **new** immutableStudent("abhi", address);

System.***out***.println("original student:\n "+s);

address.setCity("delhi");

System.***out***.println("\nAfer changing address object using local reference variable address, student becomes:\n"+ s);

// address.setCity("delhi");

s.getAddress().setCity("delhi");

System.***out***.println("\nAfer changing address object using local reference variable address, student becomes:\n"+ s);

System.***out***.println("\n Class immutableStudent is immutable");

}

}