

The code is :

```
class F implements Cloneable{
    private int f=0;    // Helping functions
    private void trace(String s) {
        System.out.println(s);
    }

    // Manager functions
    public F(int f1) { f = f1; }

    //Access functions
    public int getF() { return f; }
    public void setF(int f1) { f = f1; }

    public boolean isLargeValue(){
        return f>100;
    }

    //Implement functions
    public void changeToZero(){
        f = 0;
    }

    public void changeNumber(int num){
        f = num;
    }

    public Object clone()
    {
        try
        { return super.clone();
        }
        catch (CloneNotSupportedException e)
        {
            // This shouldn't happen, since we are Cloneable
            return null;
        }
    }
}
```

```

    public boolean equals(Object obj) {
        F tstF;

        if (!(obj instanceof F)) return false;
        tstF = (F) obj;

        return (f == tstF.f);

    }
    public String toString() {
        return ("");
    }
}

class B implements Cloneable{
    // Attributes
    private int b=0;

    // Manager functions
    public B(int b1) { b = b1; }

    // Helping functions
    private void trace(String s) {
        System.out.println(s);
    }

    //Access functions
    public int getB() { return b; }
    public void setB(int b1) { b = b1; }
    public boolean isLargeValue(){
        return b>100;
    }

    //Implement functions
    public void changeToZero(){
        b = 0;
    }

    public void changeNumber(int num){
        b = num;
    }
}

```

```

public Object clone()
{
    try
    { return super.clone();
    }
    catch (CloneNotSupportedException e)
    {
        // This shouldn't happen, since we are Cloneable
        return null;
    }
}

public boolean equals(Object obj) {
    B tstB;

    if (!(obj instanceof B)) return false;
    tstB = (B) obj;

    return (b == tstB.b);
}

public String toString() {
    return ("");
}
}

class A implements Cloneable{
    // Attributes
    private String a;
    private F fObj = new F(0);
    private B bObj = new B(0);

    // Manager functions
    public A(String a1, int f1, int b1) { a = a1; fObj.setF(f1); bObj.setB(b1); }
    // Helping functions
    private void trace(String s) {
        System.out.println(s);
    }

    //Access functions
    public B getBObj() { return bObj; }
    public void setBObj(int b1) { bObj.setB(b1); }
    public F getFObj() { return fObj; }
    public void setFObj(int f1) { fObj.setF(f1); }

    public String getA() { return a; }

```

```

        public void setA(String a1) { a = a1; }
        public boolean isBruceLee(){
            return a == "Bruce Lee";
        }
        //Implement functions
        public void changeName(String new_name){
            a = new_name;
        }

        public Object clone()
        {
            try
            {
                A a = (A)super.clone();
                a.bObj = (B)bObj.clone();
                a.fObj = (F)fObj.clone();
                return a;
            }
            catch (CloneNotSupportedException e)
            {
                // This shouldn't happen, since we are Cloneable
                return null;
            }
        }
        public boolean equals(Object obj) {
            A tstA;

            if (!(obj instanceof A)) return false;
            tstA = (A) obj;

            return (a == tstA.a && bObj.equals(tstA.bObj) && fObj.equals(tstA.fObj));
        }
        public String toString() {
            return ("");
        }
    }

    class D implements Cloneable {
        // Attributes
        private int d=0;

        // Manager functions
        public D(int d1) { d = d1; }
        // Helping functions
        private void trace(String s) {

```

```

        System.out.println(s);
    }

    //Access functions

    public int getD() { return d; }
    public void setD(int d1) { d = d1; }

    public boolean isLargeValue(){
        return d>100;
    }

    //Implement functions
    public void changeToZero(){
        d = 0;
    }
    public void changeNumber(int num){
        d = num;
    }

    public Object clone()
    {
        try
        {
            return super.clone();
        }
        catch (CloneNotSupportedException e)
        {
            // This shouldn't happen, since we are Cloneable
            return null;
        }
    }
    public String toString() {
        return ("");
    }
}

class E extends A implements Cloneable
{
    // Attributes
    private int e;
    private D dObj = new D(0);

    // Manager functions

```

```

public E(int f1,int e1, String a1, int b1, int d1) {
    super(a1,b1,f1);
    e = e1;
    dObj.setD(d1);
}

// Helping functions
private void trace(String s) {
    System.out.println(s);
}

//Access functions

public int getE() {return e;}
public void setE(int e1){e=e1;}

public D getDObj() { return dObj; }
public void setDObj(int d1) { dObj.setD(d1); }

public String getA() { return super.getA(); }
public int getB() {return getBObj().getB(); }
public int getF() {return getFObj().getF(); }

public void setA(String a1) { super.setA(a1); }
public void setB(int b1) { getBObj().setB(b1); }
public void setF(int f1) { getFObj().setF(f1); }
public boolean isLargeValue(){
    return e>100;
}

//Implement functions
public void changeToZero(){
    e = 0;
}
public void changeNumber(int num){
    e = num;
}

public boolean equals(Object obj) {
    E tstE;
    if (!(obj instanceof E)) return false;
    tstE = (E) obj;

    return (e == tstE.e);
}

```

```

    public String toString() {
        return (super.toString() + "" );
    }
    public Object clone()
    {
        return super.clone();
    }
}

public class Demo1 {
    public static void main (String argv[])
    {
        E eObj = new E(1,3,"Jack",2,4);
        System.out.println("Value of f: "+eObj.getF());
        System.out.println("Value of b: "+eObj.getB());
        System.out.println("Value of a: "+eObj.getA());
        System.out.println("Value of e: "+eObj.getE());
        System.out.println("Value of d: "+eObj.getDObj().getD());

        eObj.getBObj().changeNumber(4);
        eObj.getFObj().changeNumber(5);
        eObj.changeNumber(6);
        eObj.getDObj().changeNumber(7);
        System.out.println("\nChange the values-> \n");
        System.out.println("Value of f: "+eObj.getF());
        System.out.println("Value of b: "+eObj.getB());
        System.out.println("Value of a: "+eObj.getA());
        System.out.println("Value of e: "+eObj.getE());
        System.out.println("Value of d: "+eObj.getDObj().getD());

        E eObj1=(E)eObj.clone();
        System.out.println("\nGet New One ");
        System.out.println("Value of f: "+eObj.getF());
        System.out.println("Value of b: "+eObj.getB());
        System.out.println("Value of a: "+eObj.getA());
        System.out.println("Value of e: "+eObj.getE());
        System.out.println("Value of d: "+eObj.getDObj().getD());

        if (eObj1.equals(eObj)){
            System.out.println("\nNew one is equal");
        }
        else

```

```

    {
        System.out.println("\nNew one is not equal");
    }
}
}

```

The screenshot shows an IDE window with a Java code editor, an execution configuration panel, and a console output window.

Code Editor: The code is a Java program that prints the values of variables f, b, a, e, and d. It then compares the values of f and b using the equals method. If they are equal, it prints "New one is equal"; otherwise, it prints "New one is not equal".

```

293 System.out.println("\nNew one ");
294 System.out.println("Value of f: "+eObj.getF());
295 System.out.println("Value of b: "+eObj.getB());
296 System.out.println("Value of a: "+eObj.getA());
297 System.out.println("Value of e: "+eObj.getE());
298 System.out.println("Value of d: "+eObj.getDObj().getD());
299
300 if (eObj1.equals(eObj)){
301     System.out.println("\nNew one is equal");
302 }
303 else
304 {
305     System.out.println("\nNew one is not equal");
306 }
307 }
308 }
309

```

Execute Mode, Version, Inputs & Arguments: The configuration panel shows the JDK version as 11.0.4. The "Interactive" checkbox is checked. The "Stdin Inputs" field is empty. The "CommandLine Arguments" field is also empty.

Execute: A blue button with a play icon is used to execute the code.

Result: The output window shows the results of the execution. It displays the values of f, b, a, e, and d, followed by the comparison result.

```

CPU Time: 0.26 sec(s), Memory: 34952 kilobyte(s) compiled and executed in 0.993 sec(s)

Value of f: 2
Value of b: 1
Value of a: Jack
Value of e: 3
Value of d: 4

Change the values->

Value of f: 5
Value of b: 4
Value of a: Jack
Value of e: 6
Value of d: 7

Get New One
Value of f: 5
Value of b: 4
Value of a: Jack
Value of e: 6
Value of d: 7

New one is equal

```

The Windows taskbar at the bottom shows the time as 17:58 on 2020/10/11.