The code is :

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


    // Manager functions
    public A(int a1) { a = a1; }

    //Access functions
    public int getA() { return a; }
    public void setA(int a1) { a = a1; }

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

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

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

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

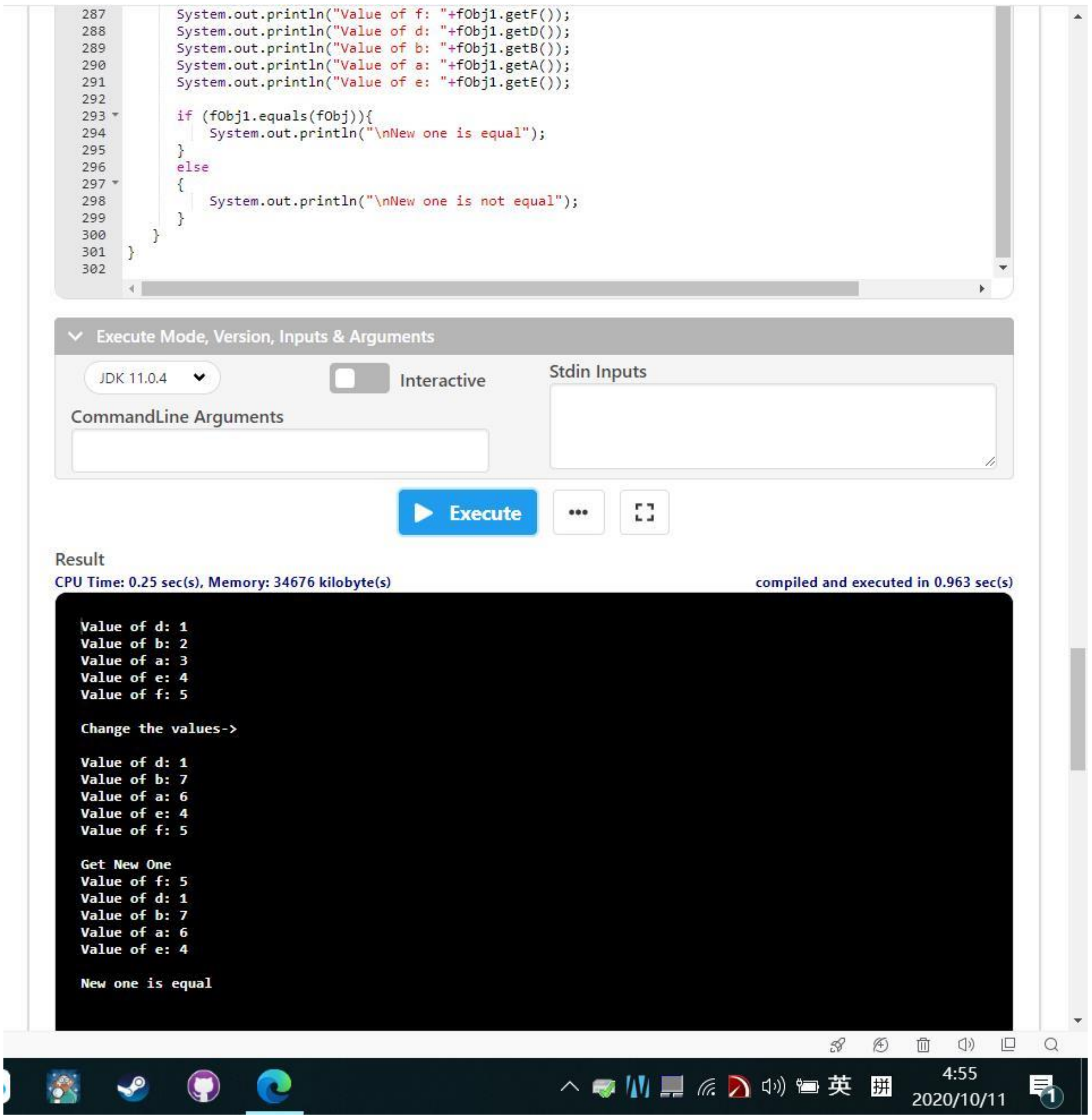
```
      public boolean equals(Object obj) {
      A tstA;

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

      return (a == tstA.a);

        }
        public String toString() {
            return ("");
      }
}
class E implements Cloneable{
        // Attributes
        private int e=0;

        // Manager functions
        public E(int e1) { e = e1; }



        // 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 boolean isLargeValue(){
            return e>100;
      }

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

      public void changeNumber(int num){
            e = 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) {
            E tstE;

            if (!(obj instanceof E)) return false;
            tstE = (E) obj;

            return (e == tstE.e);
        }
        public String toString() {
          return ("");
      }
}

class B implements Cloneable{
          // Attributes
          private int b;
          private A aObj = new A(0);
          private E eObj = new E(0);

          // Manager functions
          public B(int b1, int a1, int e1) { b = b1; aObj.setA(a1); eObj.setE(e1);}
          // Helping functions
          private void trace(String s) {
              System.out.println(s);
          }

          //Access functions
          public A getAObj() { return aObj; }
          public void setAObj(int a1) { aObj.setA(a1); }
          public E getEObj() { return eObj; }
          public void setEObj(int e1) { eObj.setE(e1); }

          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
              {
                B b = (B)super.clone();
                b.aObj = (A)aObj.clone();
                b.eObj = (E)eObj.clone();
                return b;
              }
              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 && aObj.equals(tstB.aObj) && eObj.equals(tstB.eObj));
        }
        public String toString() {
            return ("");
      }
}
class D implements Cloneable {
        // Attributes
        private int d=0;
        private B bObj = new B(0,0,0);
```

```
    // Manager functions
    public D(int d1, int b1, int a1, int e1) { d = d1; bObj.setB(b1); bObj.getAObj().setA(a1);
bObj.getEObj().setE(e1);}
    // 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 A getAObj() { return bObj.getAObj(); }
    public void setAObj(int a1) { bObj.getAObj().setA(a1); }
    public E getEObj() { return bObj.getEObj(); }
    public void setEObj(int e1) { bObj.getEObj().setE(e1); }

    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
        {
           D d = (D)super.clone();
           d.bObj = (B)bObj.clone();
           return d;
        }
        catch (CloneNotSupportedException e)
        {
               // This shouldn't happen, since we are Cloneable
```

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

class F extends D implements Cloneable
{
            // Attributes
            private int f;

            // Manager functions
            public F(int d1, int b1, int a1, int e1, int f1) {
                    super(d1,b1,a1,e1);
                    f = f1;

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

            //Access functions

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

            public int getD() { return super.getD(); }
            public int getB() {return getBObj().getB(); }
            public int getA() {return getBObj().getAObj().getA(); }
            public int getE() {return getBObj().getEObj().getE(); }

            public void setD(int d1) { super.setD(d1); }
            public void setB(int b1) { getBObj().setB(b1); }
            public void setA(int a1) { getBObj().getAObj().setA(a1); }
            public void setE(int e1) { getBObj().getEObj().setE(e1); }
            public boolean isLargeValue(){
                  return f>100;
            }

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

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

    public boolean equals(Object obj) {
      F tstA;
      if (!(obj instanceof F)) return false;
      tstA = (F) obj;

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

}

public class Demo1 {
      public static void main (String argv[])
      {
            F fObj = new F(1,2,3,4,5);
            System.out.println("Value of d: "+fObj.getD());
            System.out.println("Value of b: "+fObj.getB());
            System.out.println("Value of a: "+fObj.getA());
            System.out.println("Value of e: "+fObj.getE());
            System.out.println("Value of f: "+fObj.getF());

            fObj.getBObj().changeNumber(7);
            fObj.getBObj().getAObj().changeNumber(6);
            System.out.println("\nChange the values-> \n");
            System.out.println("Value of d: "+fObj.getD());
            System.out.println("Value of b: "+fObj.getB());
            System.out.println("Value of a: "+fObj.getA());
            System.out.println("Value of e: "+fObj.getE());
            System.out.println("Value of f: "+fObj.getF());
            F fObj1=(F)fObj.clone();
            System.out.println("\nGet New One ");
            System.out.println("Value of f: "+fObj1.getF());
            System.out.println("Value of d: "+fObj1.getD());
            System.out.println("Value of b: "+fObj1.getB());
            System.out.println("Value of a: "+fObj1.getA());
            System.out.println("Value of e: "+fObj1.getE());

            if (fObj1.equals(fObj)){
                System.out.println("\nNew one is equal");
            }
            else
            {
                System.out.println("\nNew one is not equal");
            }
        }
    }
```


```
        System.out.println("Value of f: "+fObj1.getF());
        System.out.println("Value of d: "+fObj1.getD());
        System.out.println("Value of b: "+fObj1.getB());
        System.out.println("Value of a: "+fObj1.getA());
        System.out.println("Value of e: "+fObj1.getE());

        if (fObj1.equals(fObj)){
            System.out.println("\nNew one is equal");
        }
        else
        {
            System.out.println("\nNew one is not equal");
        }
    }
}

```

CPU Time: 0.25 sec(s), Memory: 34676 kilobyte(s)   compiled and executed in 0.963 sec(s)

```
Value of d: 1
Value of b: 2
Value of a: 3
Value of e: 4
Value of f: 5

Change the values->

Value of d: 1
Value of b: 7
Value of a: 6
Value of e: 4
Value of f: 5

Get New One
Value of f: 5
Value of d: 1
Value of b: 7
Value of a: 6
Value of e: 4

New one is equal
```