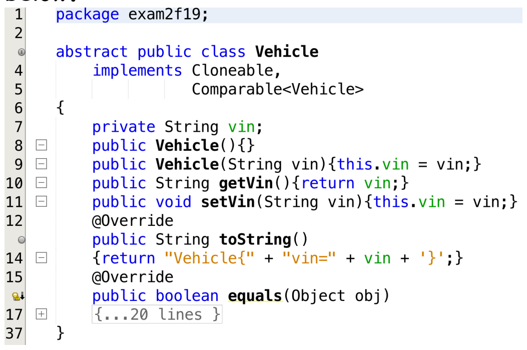
South Louisiana Community College
ASDV 2420, Advanced Programming Language I
Programming Examination 2 on 2019/3/12

Create a project called Exam2LastName where LastName is your last name.
Upload the zip and html for each problem outside of the zip. The examination MUST be uploaded before 11:59am. IT WILL CLOSE at 11:59am.

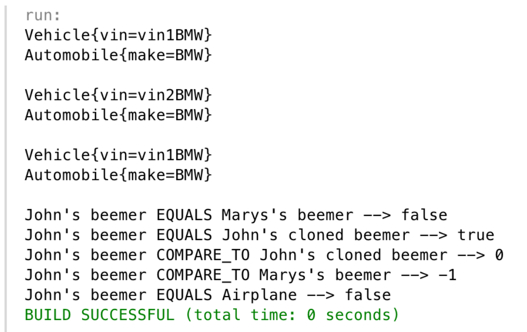
Problem 1 (4 points)
Create a the *abstract* class <u>Vehicle</u> which implements <u>Cloneable</u> and <u>Comparable.</u>
You are NOT to implement <u>Cloneable</u> and <u>Comparable</u> inside this abstract class.
Use Netbeans to insert code for the following: The class methods are shown below.

Problem 2 (3 points)
Create a the class *<u>Automobile</u>*. which extends *<u>Vehicle</u>* implements *<u>Cloneable</u>* *as shown in RHS.* HERE you implement *<u>clone()</u>* *and* <u>comparedTo</u> <u>methods.</u> The <u>compareTo</u> compares vin numbers, nothing else. ***TEST* your CODE with the <u>main</u> shown at the RHS to produce the exact output shown below.**

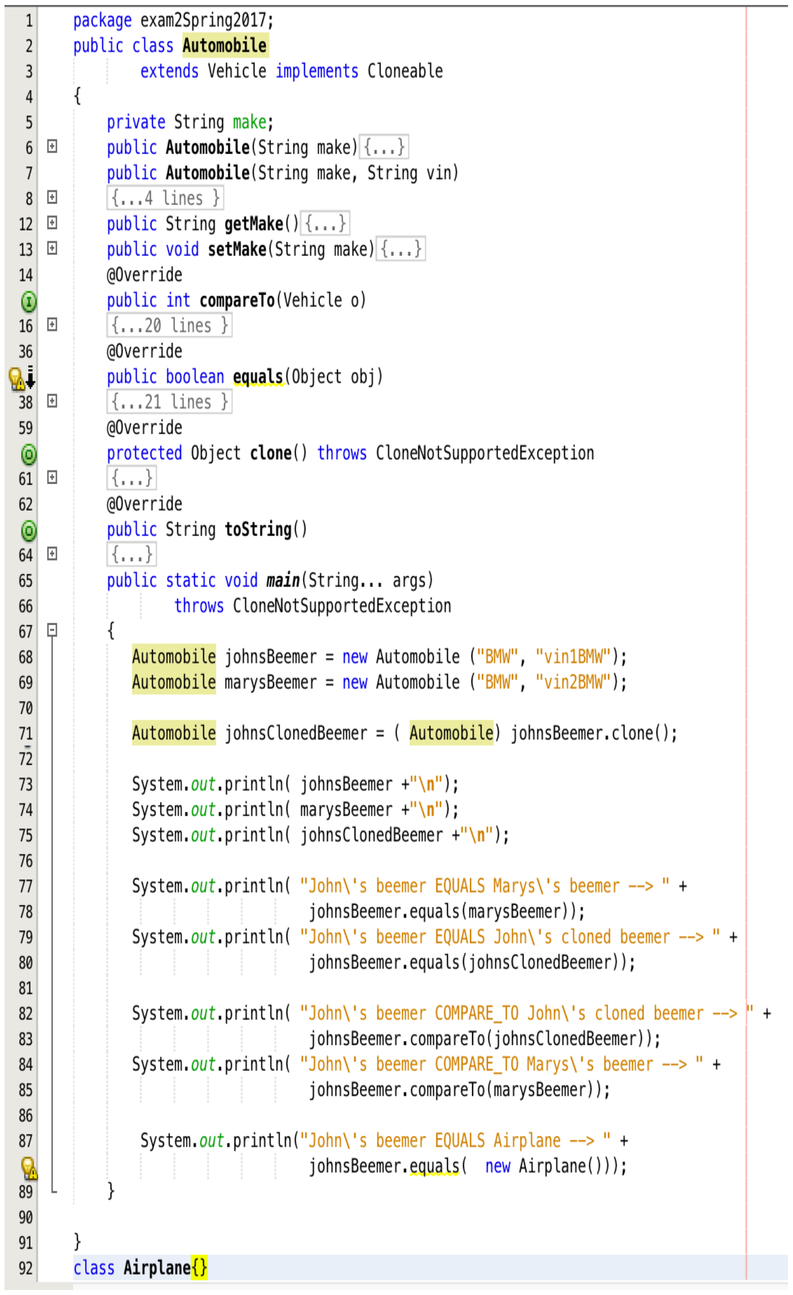


```
package exam2f19;

abstract public class Vehicle
    implements Cloneable,
               Comparable<Vehicle>
{
    private String vin;
    public Vehicle(){}
    public Vehicle(String vin){this.vin = vin;}
    public String getVin(){return vin;}
    public void setVin(String vin){this.vin = vin;}
    @Override
    public String toString()
    {return "Vehicle{" + "vin=" + vin + '}';}
    @Override
    public boolean equals(Object obj)
    {...20 lines }
}
```

OUTPUT



```
run:
Vehicle{vin=vin1BMW}
Automobile{make=BMW}

Vehicle{vin=vin2BMW}
Automobile{make=BMW}

Vehicle{vin=vin1BMW}
Automobile{make=BMW}

John's beemer EQUALS Marys's beemer --> false
John's beemer EQUALS John's cloned beemer --> true
John's beemer COMPARE_TO John's cloned beemer --> 0
John's beemer COMPARE_TO Marys's beemer --> -1
John's beemer EQUALS Airplane --> false
BUILD SUCCESSFUL (total time: 0 seconds)
```

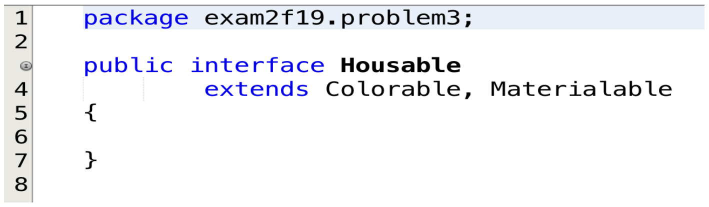


```
package exam2Spring2017;
public class Automobile
        extends Vehicle implements Cloneable
{
    private String make;
    public Automobile(String make) {...}
    public Automobile(String make, String vin)
    {...4 lines }
    public String getMake() {...}
    public void setMake(String make) {...}
    @Override
    public int compareTo(Vehicle o)
    {...20 lines }
    @Override
    public boolean equals(Object obj)
    {...21 lines }
    @Override
    protected Object clone() throws CloneNotSupportedException
    {...}
    @Override
    public String toString()
    {...}
    public static void main(String... args)
            throws CloneNotSupportedException
    {
        Automobile johnsBeemer = new Automobile ("BMW", "vin1BMW");
        Automobile marysBeemer = new Automobile ("BMW", "vin2BMW");

        Automobile johnsClonedBeemer = ( Automobile) johnsBeemer.clone();

        System.out.println( johnsBeemer +"\n");
        System.out.println( marysBeemer +"\n");
        System.out.println( johnsClonedBeemer +"\n");

        System.out.println( "John\'s beemer EQUALS Marys\'s beemer --> " +
                            johnsBeemer.equals(marysBeemer));
        System.out.println( "John\'s beemer EQUALS John\'s cloned beemer --> " +
                            johnsBeemer.equals(johnsClonedBeemer));

        System.out.println( "John\'s beemer COMPARE_TO John\'s cloned beemer --> " +
                            johnsBeemer.compareTo(johnsClonedBeemer));
        System.out.println( "John\'s beemer COMPARE_TO Marys\'s beemer --> " +
                            johnsBeemer.compareTo(marysBeemer));

         System.out.println("John\'s beemer EQUALS Airplane --> " +
                            johnsBeemer.equals(   new Airplane()));
    }

}
class Airplane{}
```

Problem 3 (2 points)
1. Create an **interface** named _Materialable_ which has one parameterless method named _material_ that returns _void._

2. Create another **interface** named _Colorable_ which has one parameterless method named _color_ that returns _void._

3. Create a third **interface** named _Housable_ which inherits all methods from interfaces _Materialable_ and _Colorable ,_

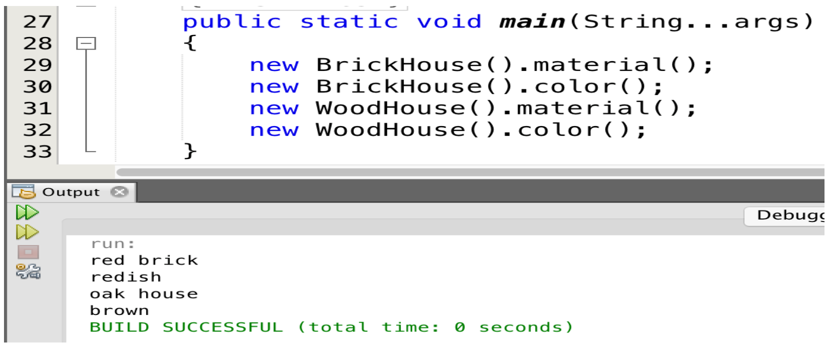


```java
package exam2f19.problem3;

public interface Housable
        extends Colorable, Materialable
{

}
```

4. Create a **class** named _BrickHouse_ which inherits all methods from interface _Housable._

5. Create a **class** named _WoodHouse_ which inherits all methods from interface _Housable._

6. Add appropriate code in the methods of classes _BrickHouse_ and _WoodHouse_ to produce the same output shown below using the _main_ method given below.

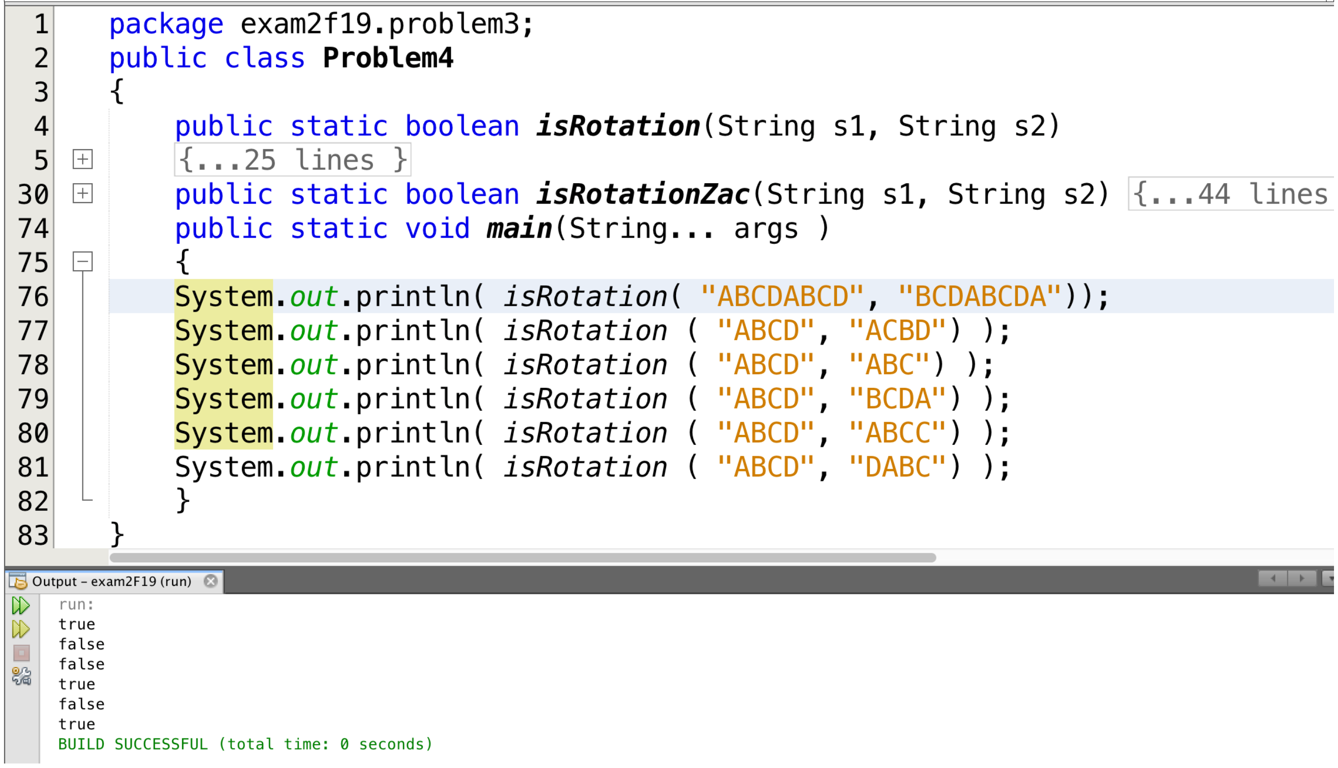


```java
public static void main(String...args)
{
    new BrickHouse().material();
    new BrickHouse().color();
    new WoodHouse().material();
    new WoodHouse().color();
}
```

```
run:
red brick
redish
oak house
brown
BUILD SUCCESSFUL (total time: 0 seconds)
```

Problem 4 (1 point)
Given any string s1 and any string s2, of any size, implement method *isRotation* shown to the right.

the method *isRotation* returns true if s2 is a rotation of s1.
(eg given s1 = ABCD and s2 = CDAB, returns true,
ABCD and s2 = DABC, returns true,
given s1 = ABCD, and s2 = ACBD , returns false
given s1 = "", and s2 = "A" , returns false)
Test it with the exact main shown below:



```
package exam2f19.problem3;
public class Problem4
{
    public static boolean isRotation(String s1, String s2)
    {...25 lines }
    public static boolean isRotationZac(String s1, String s2) {...44 lines
    public static void main(String... args )
    {
    System.out.println( isRotation( "ABCDABCD", "BCDABCDA"));
    System.out.println( isRotation ( "ABCD", "ACBD") );
    System.out.println( isRotation ( "ABCD", "ABC") );
    System.out.println( isRotation ( "ABCD", "BCDA") );
    System.out.println( isRotation ( "ABCD", "ABCC") );
    System.out.println( isRotation ( "ABCD", "DABC") );
    }
}
```

Output - exam2F19 (run)

```
run:
true
false
false
true
false
true
BUILD SUCCESSFUL (total time: 0 seconds)
```