

FMSS Bootcamp 2022

3.Hafta

String içinde neden dönülemez ?

- for ({*Niteleyici*} *Tür değişken: İfade*) Statement
- İfadenin türü, Iterable türünün bir alt türü veya bir dizi türü olmalıdır, aksi takdirde bir derleme zamanı hatası oluşur.
- Gelişmiş for ifadesinin kesin anlamı, aşağıdaki gibi bir temel for ifadesine çevrilerek verilir:

```
for (I #i = Expression.iterator(); #i.hasNext(); ) {  
    {VariableModifier} TargetType Identifier =  
        (TargetType) #i.next();  
    Statement  
}
```

Başlangıçta ifadenin Iterator nesnesi alınır.
Ardından her adımda bu nesnenin hasNext()
koşuluna bakılır ve next() ifadesinin sonucu
ele alınır.

String neden iterable değil ?

- Bu soru için aldığım en tatmin edici cevap şu:
- Eğer string iterable olsaydı, her bir karakteri char'dan Character'e dönüştürmemiz gerekirdi (boxing). Bu da maliyetli bir işlem olduğu için tercih edilmemiştir.
- https://github.com/burhan-takesen/FMSS_2022/blob/main/FMSS/src/thirdweek/IterableString.java (iterable string ??)

Kaynaklar

- <https://docs.oracle.com/javase/specs/jls/se11/html/jls-14.html#jls-14.14.2>

Neden Final Java 8'den
Sonra Daha Az
Kullanılıyor

Nested Sınıf Nedir?

```
class OuterClass {  
    ...  
    class NestedClass {  
        ...  
    }  
}
```

Bir sınıfın içinde oluşturulan sınıflara denir. Birazdan bazı nested türleri ile final arasındaki ilişkiye bakıyor olacağız.

Local Sınıflar Nedir?

Bir sınıfın içindeki bloklardan birinde tanımlanan bir tür nested sınıftır.

```
3  class Outer {
4
5      // a Class inside Constructor
6      Outer(){
7
8          class ConstructorLocal {
9
10         }
11     }
12
13     // a class inside instance block
14     {
15         class Local{
16
17         }
18     }
19
20     // a class inside static block
21     static {
22         class StaticLocal{
23
24         }
25     }
26
27     // a class inside method
28     public void add(int a, int b) {
29
30         class MethodLocal {
31
32         }
33
34         // a class inside if block
35         if(a > b) {
36
37             class Local{
38
39             }
40         }
41     }
42 }
```

Anonim Sınıflar Nedir?

Bir sınıfın içinde sınıf ismi verilmeden oluşturulan bir tür nested sınıftır.

```
class outerClass {  
  
    // defining anonymous class  
    object1 = new Type(parameterList) {  
        // body of the anonymous class  
    };  
}
```

Local ve Anonim Sınıflar ile Final Bağlantısı

Java 8'den önce local ve anonim sınıflar içerisinde daha üst scope'tan bir değişkene ulaşmamız için bu değişkenlerin final olması gerekiyordu.

Java 8'den Önce Local Sınıf ve Final

```
3 public class Example {  
4  
5     public static void aMethod(){  
6  
7         final int myInteger = 11;  
8  
9         class LocalClass {  
10  
11             int returnMyInteger() {  
12  
13                 return myInteger;  
14             }  
15         }  
16     }  
17 }
```

```
3 public class Example {  
4  
5     public static void aMethod(){  
6  
7         int myInteger = 11;  
8  
9         class LocalClass {  
10  
11             int returnMyInteger() {  
12  
13                 return myInteger; // error before Java 8  
14             }  
15         }  
16     }  
17 }
```

Java 8'den Önce Anonim Sınıf ve Final

```
45 public class Example {  
46  
47     public void method(){  
48  
49         final String englishGreeting = "Hello";  
50  
51         Runnable r = new Runnable() {  
52             @Override  
53             public void run() {  
54                 System.out.println(englishGreeting);  
55             }  
56         };  
57     }  
58 }
```

```
47 public void method(){  
48  
49     String englishGreeting = "Hello";  
50  
51     Runnable r = new Runnable() {  
52         @Override  
53         public void run() {  
54             System.out.println(englishGreeting);  
55             //error before Java 8  
56         }  
57     };  
58 }  
59 }
```

Effectively Final Nedir?

Final keyword kullanılmadan tanımlanan ve sadece 1 kez atama yapılan değişkenlere denir.

Effectively Final Örnekleri

```
3 public class Example {
4
5     public static void aMethod(){
6
7         int myInteger = 11;
8
9         class LocalClass {
10
11             int returnMyInteger() {
12
13                 return myInteger; //runs after Java 8
14             }
15         }
16     }
17 }
```

```
47 public class Example {
48
49     public void method(){
50
51         String englishGreeting = "Hello";
52
53         Runnable r = new Runnable() {
54             @Override
55             public void run() {
56                 System.out.println(englishGreeting);
57                 //runs after Java 8
58             }
59         };
60
61     }
62 }
```

Effectively Final Hatalı Kullanımı

```
3 public class Example {
4
5     public static void aMethod(){
6
7         int myInteger = 11;
8
9         class LocalClass {
10
11             int returnMyInteger() {
12
13                 return myInteger; //error
14             }
15         }
16
17         myInteger++;
18     }
19 }
```

```
47 public class Example {
48
49     public void method(){
50
51         String englishGreeting = "Hello";
52
53         Runnable r = new Runnable() {
54             @Override
55             public void run() {
56                 System.out.println(englishGreeting); //error
57             }
58         };
59
60         englishGreeting = "Hi";
61     }
62 }
```



```
// Call by value
int a = 5;
int b = 10;
System.out.println("a:" + a + " b:" + b);
swap(a, b);    //swap(5, 10)
System.out.println("a:" + a + " b:" + b);
```

CallBy ×

C:\Users\Emrullah7\.jdk\openjdk-18.0.2\bin\java.exe "-javaagent:

a:5 b:10

a:5 b:10

Process finished with exit code 0

```
static void swap(int a, int b){
```

```
    // primitive tiplerde call by value geçerlidir.
```

```
    int temp = a;
```

```
    a = b;
```

```
    b = temp;
```

```
}
```

```

class Customer {

    4 usages
    String name;
    4 usages
    String lastName;
    4 usages
    int age;

    // constructor
    // getter
    // setter
    // toString
}

```

```

static void swap2(Customer customer1, Customer customer2){

    // referans tiplerde call by value geçerlidir.
    Customer temp = customer1;
    customer1 = customer2;
    customer2 = temp;

    // customer1 = 0x103
    // customer2 = 0x102
    // buradaki değerler scope bitene kadar geçerlidir.
}

```

```

Customer customer1 = new Customer( name: "Ahmet", lastName: "Yilmaz", age: 26);
Customer customer2 = new Customer( name: "Mehmet", lastName: "Yilmaz", age: 33);
System.out.println(customer1);
swap2(customer1, customer2); //swap2(0x102, 0x103)
System.out.println(customer1);

```

CallBy x

```

C:\Users\Emrullah7\.jdk\openjdk-18.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Intelli
Customer{name='Ahmet', lastName='Yilmaz', age=26}
Customer{name='Ahmet', lastName='Yilmaz', age=26}

```

```
// Call by reference
Customer customer3 = new Customer( name: "Ahmet", lastName: "Yilmaz", age: 26);
Customer customer4 = new Customer( name: "Mehmet", lastName: "Yilmaz", age: 33);
System.out.println(customer3);
swap3(customer3, customer4); //swap3(0x102, 0x103)
System.out.println(customer3);
```

CallBy ×

```
C:\Users\Emrullah7\.jdk\openjdk-18.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ
Customer{name='Ahmet', lastName='Yilmaz', age=26}
Customer{name='Mehmet', lastName='Yilmaz', age=26}
```

```
static void swap3(Customer customer1, Customer customer2){

    //referans tiplerin gösterdiği objelerde call by reference geçerlidir.
    String temp = customer1.getName();
    customer1.setName(customer2.getName());
    customer2.setName(temp);

    // customer1 = 0x102.name = "Mehmet"
}
```

```
Customer customer5 = new Customer( name: "Ahmet", lastName: "Yilmaz", age: 26);  
Customer customer6 = new Customer( name: "Mehmet", lastName: "Yilmaz", age: 33);  
System.out.println(customer5);  
swap4(customer5, customer6); //swap4(0x102, 0x103)  
System.out.println(customer5);
```

CallBy x

```
C:\Users\Emrullah7\.jdk\openjdk-18.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Int  
Customer{name='Ahmet', lastName='Yilmaz', age=26}  
Customer{name='Ahmet', lastName='Yilmaz', age=33}
```

```
static void swap4(Customer customer1, Customer customer2){  
  
    //referans tiplerin gösterdiği objelerde call by reference geçerlidir.  
    int temp = customer1.getAge();  
    customer1.setAge(customer2.getAge());  
    customer2.setAge(temp);  
  
    // customer1 = 0x102.age = 33  
}
```

**References
and
Local Method Variables**

Stack

**Classes
and
Objects**

Heap

```

public class Main {
    public static void main(String[] args) {
        Mercedes m1 = new Mercedes( name: "CLA", electrical: true , age: 2);
        Mercedes m2 = new Mercedes( name: "E200", electrical: false , age: 5);
        System.out.println("Name: " + m1.getName()+ " Age: " + m1.getAge()+ " Electrical: " + m1.isElectricVehicle());
        System.out.println("Name: " + m2.getName()+ " Age: " + m2.getAge()+ " Electrical: " + m2.isElectricVehicle());
        System.out.println(Mercedes.getSoldVehicleNumber());
    }
}

```

```

import java.math.BigDecimal;
import java.util.Random;

public class Mercedes {

    private static int soldVehicleNumber;

    public static int getSoldVehicleNumber() { return soldVehicleNumber; }

    private String name;
    private BigDecimal serialNumber;
    private int age;
    private boolean isElectricVehicle;

    public Mercedes(String name, boolean electrical, int age) {
        this.name = name;
        this.serialNumber = BigDecimal.valueOf(new Random().nextInt( bound: 1000));
        this.age = age;
        soldVehicleNumber++;
        isElectricVehicle = electrical;
    }

    public String getName() {
        return name;
    }


    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

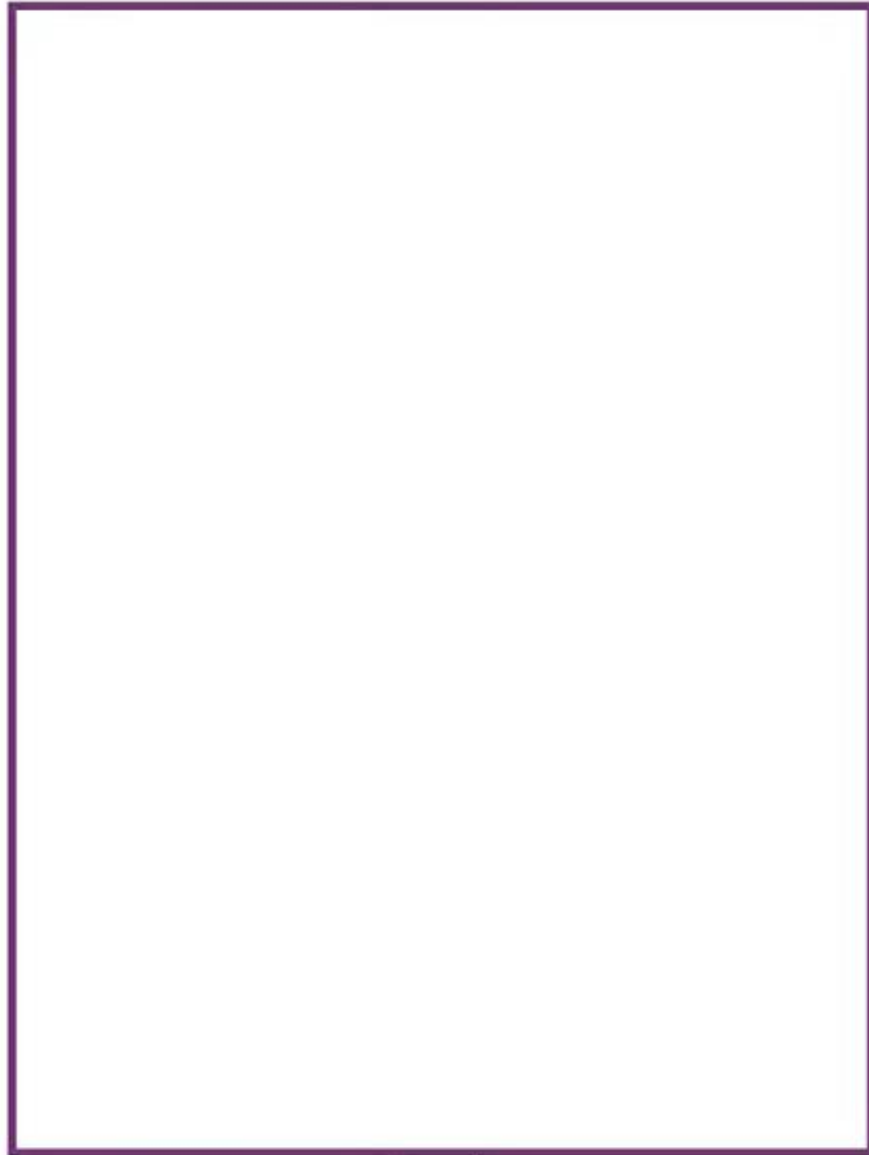
    public void setAge(int age) {
        this.age = age;
    }
}

```

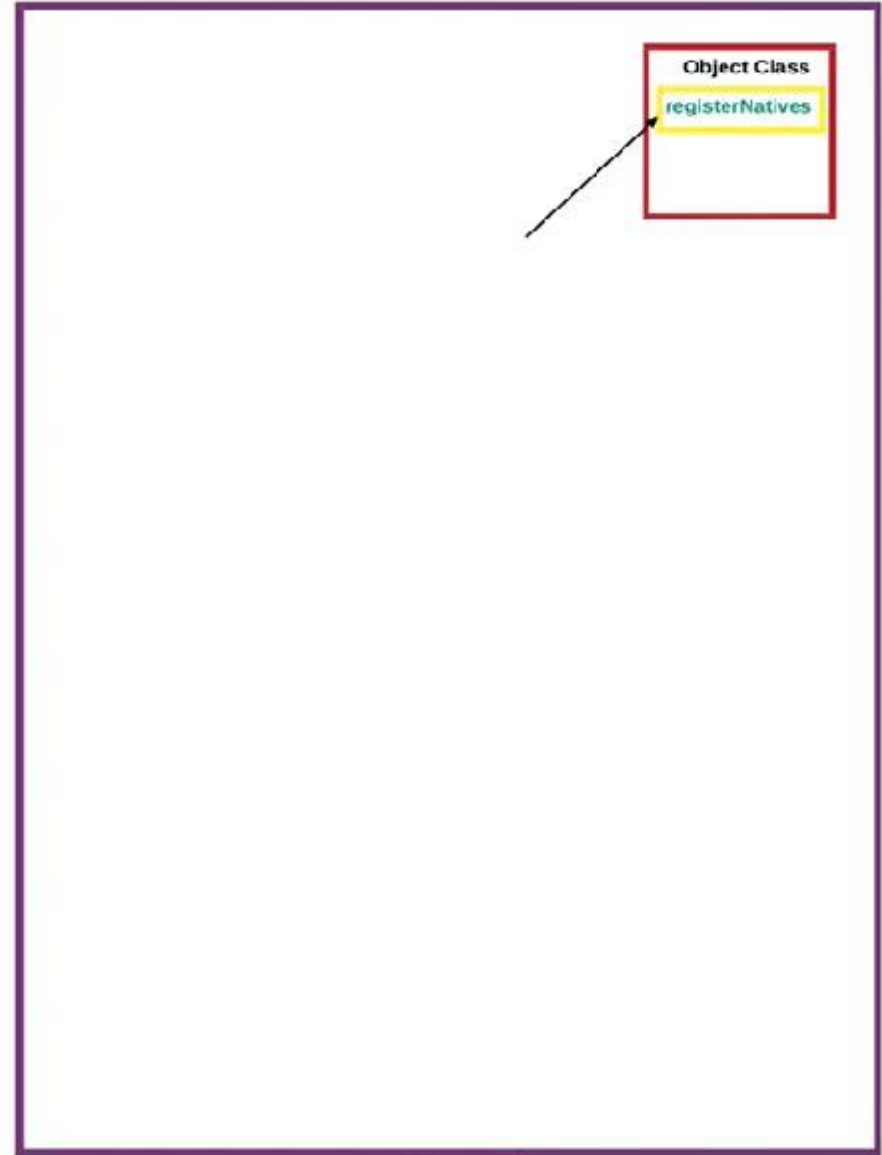
Main extends Object




```
public class Main {  
    public static void main(String[] args) {  
        Mercedes m1 = new Mercedes( name: "CLA", electrical: true , age: 2);  
        Mercedes m2 = new Mercedes( name: "E200", electrical: false , age: 5);  
        System.out.println("Name: " + m1.getName() + " Age: " + m1.getAge() + " Electrical: " + m1.isElectricVehicle());  
        System.out.println("Name: " + m2.getName() + " Age: " + m2.getAge() + " Electrical: " + m2.isElectricVehicle());  
        System.out.println(Mercedes.getSoldVehicleNumber());  
    }  
}
```



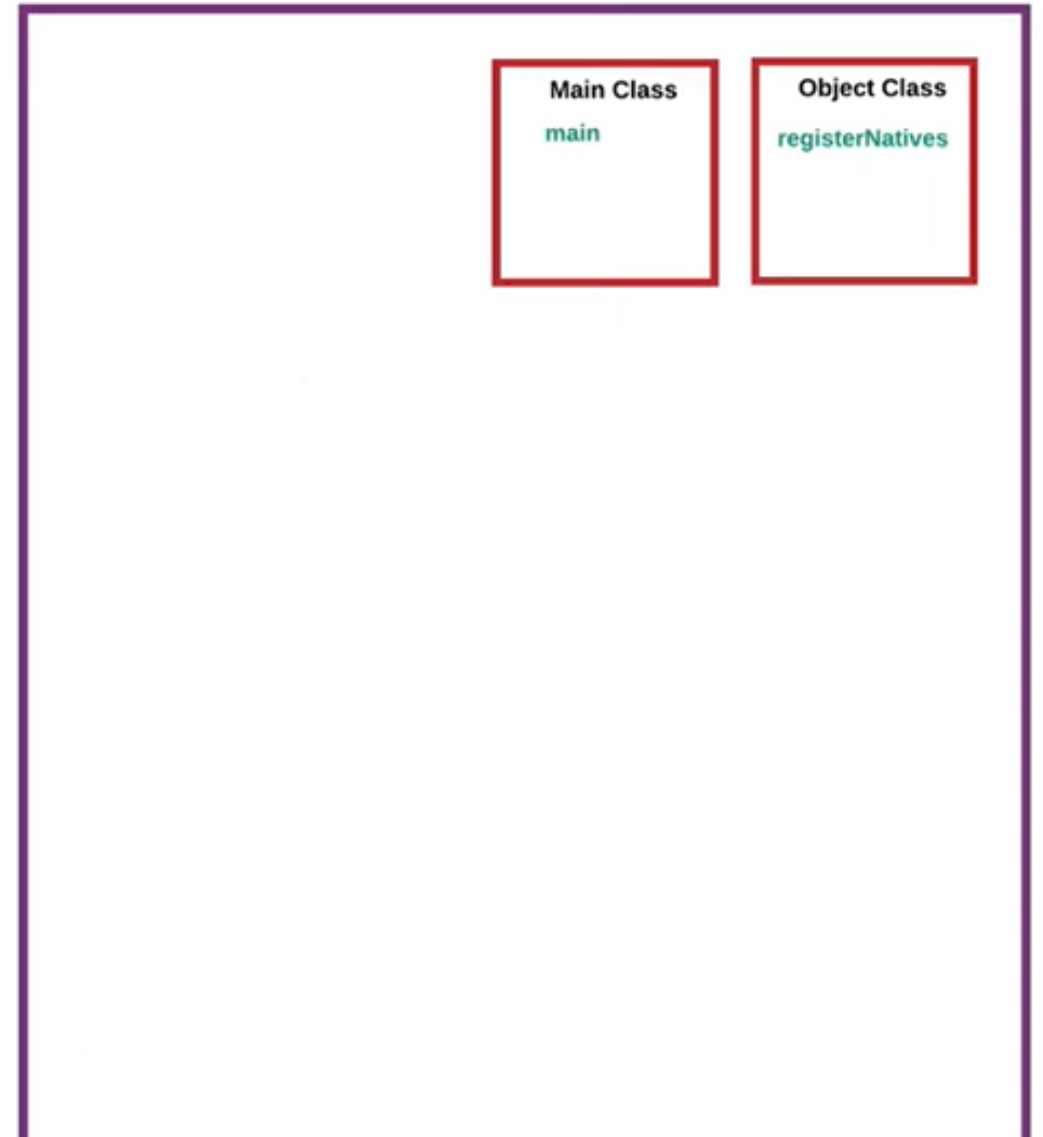
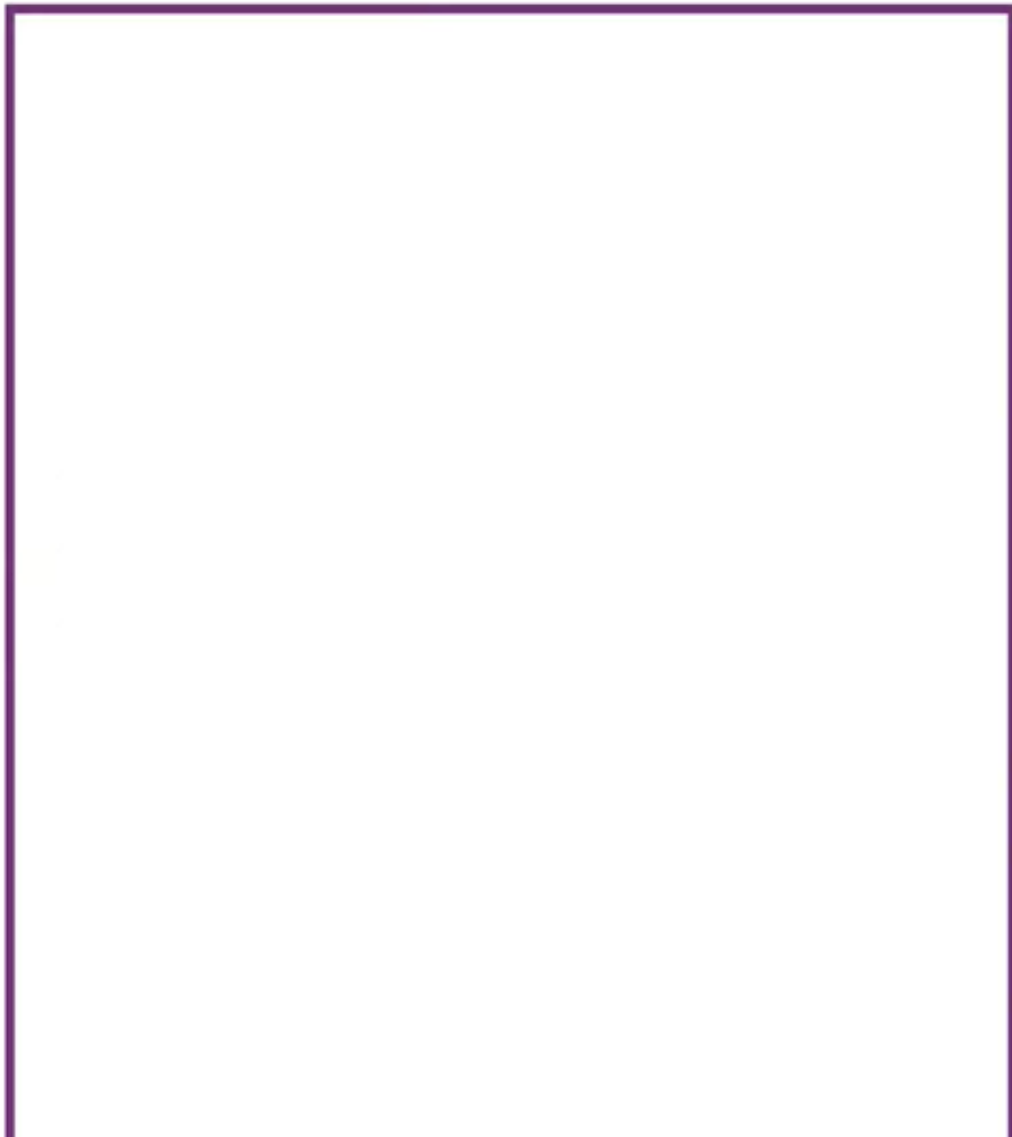
Stack



Heap



```
public class Main {  
    public static void main(String[] args) {  
        Mercedes m1 = new Mercedes( name: "CLA", electrical: true , age: 2);  
        Mercedes m2 = new Mercedes( name: "E200", electrical: false , age: 5);  
        System.out.println("Name: " + m1.getName() + " Age: " + m1.getAge() + " Electrical: " + m1.isElectricVehicle());  
        System.out.println("Name: " + m2.getName() + " Age: " + m2.getAge() + " Electrical: " + m2.isElectricVehicle());  
        System.out.println(Mercedes.getSoldVehicleNumber());  
    }  
}
```



```

public class Main {
    public static void main(String[] args) {
        Mercedes m1 = new Mercedes( name: "CLA", electrical: true , age: 2);
        Mercedes m2 = new Mercedes( name: "E200", electrical: false , age: 5);
        System.out.println("Name: " + m1.getName() + " Age: " + m1.getAge() + " Electrical: " + m1.isElectricVehicle());
        System.out.println("Name: " + m2.getName() + " Age: " + m2.getAge() + " Electrical: " + m2.isElectricVehicle());
        System.out.println(Mercedes.getSoldVehicleNumber());
    }
}

```

```

import java.math.BigDecimal;
import java.util.Random;

public class Mercedes {

    private static int soldVehicleNumber;

    public static int getSoldVehicleNumber() { return soldVehicleNumber; }

    private String name;
    private BigDecimal serialNumber;
    private int age;
    private boolean isElectricVehicle;

    public Mercedes(String name, boolean electrical, int age) {
        this.name = name;
        this.serialNumber = BigDecimal.valueOf(new Random().nextInt( bound: 1000));
        this.age = age;
        soldVehicleNumber++;
        isElectricVehicle = electrical;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public boolean isElectricVehicle() {

```

```

public class Main {
    public static void main(String[] args) {
        Mercedes m1 = new Mercedes( name: "CLA", electrical: true , age: 2);
        Mercedes m2 = new Mercedes( name: "E280", electrical: false , age: 5);
        System.out.println("Name: " + m1.getName() + " Age: " + m1.getAge() + " Electrical: " + m1.isElectricVehicle());
        System.out.println("Name: " + m2.getName() + " Age: " + m2.getAge() + " Electrical: " + m2.isElectricVehicle());
        System.out.println(Mercedes.getSoldVehicleNumber());
    }
}

```

Mercedes extends Object

```

import java.math.BigDecimal;
import java.util.Random;

public class Mercedes {

    private static int soldVehicleNumber;

    public static int getSoldVehicleNumber() { return soldVehicleNumber; }

    private String name;
    private BigDecimal serialNumber;
    private int age;
    private boolean isElectricVehicle;

    public Mercedes(String name, boolean electrical, int age) {
        this.name = name;
        this.serialNumber = BigDecimal.valueOf(new Random().nextInt( bound: 1000));
        this.age = age;
        soldVehicleNumber++;
        isElectricVehicle = electrical;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}

```

```
import java.math.BigDecimal;
import java.util.Random;

public class Mercedes {

    private static int soldVehicleNumber;

    public static int getSoldVehicleNumber() { return soldVehicleNumber; }

    private String name;
    private BigDecimal serialNumber;
    private int age;
    private boolean isElectricVehicle;

    public Mercedes(String name, boolean electrical, int age) {
        this.name = name;
        this.serialNumber = BigDecimal.valueOf(new Random().nextInt( bound: 1000));
        this.age = age;
        soldVehicleNumber++;
        isElectricVehicle = electrical;
    }

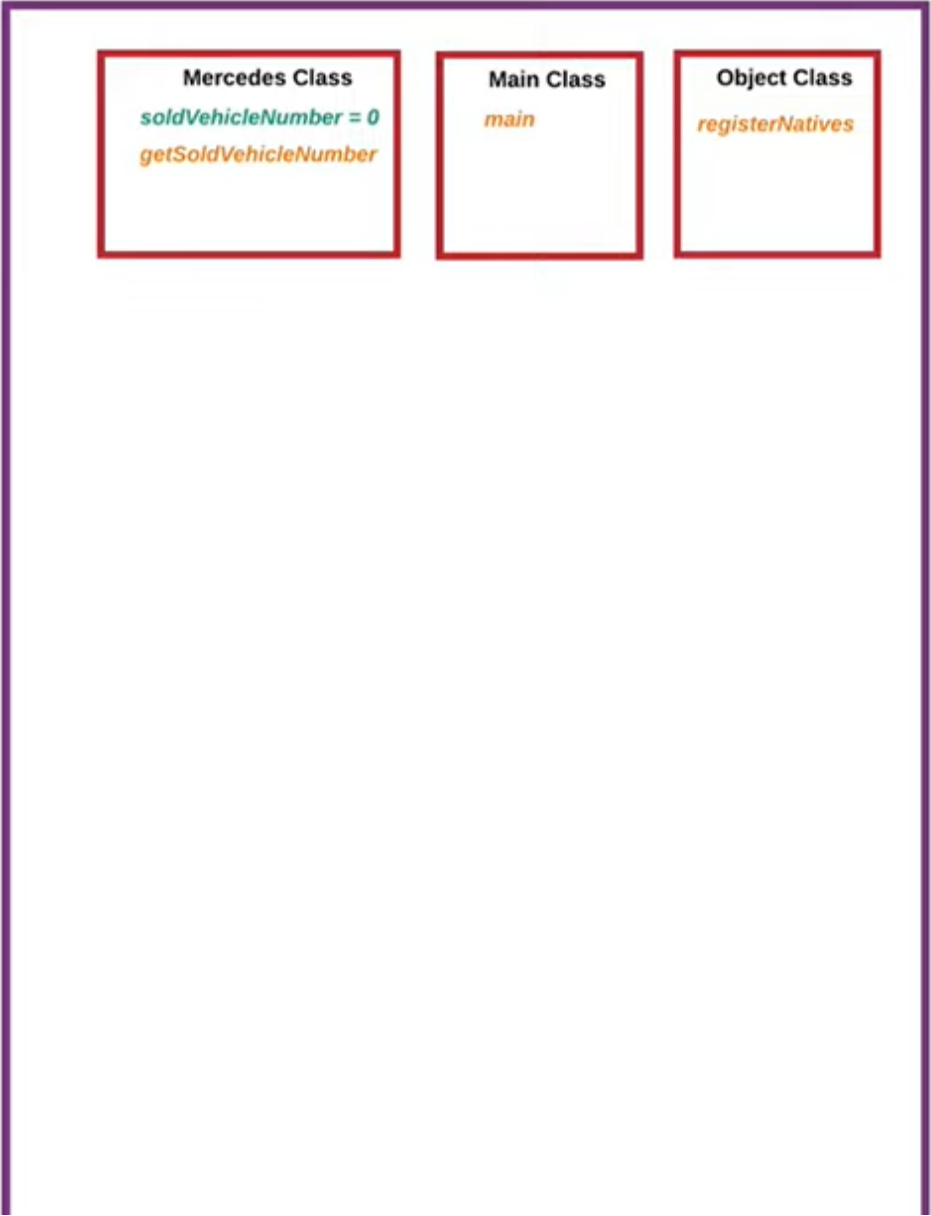
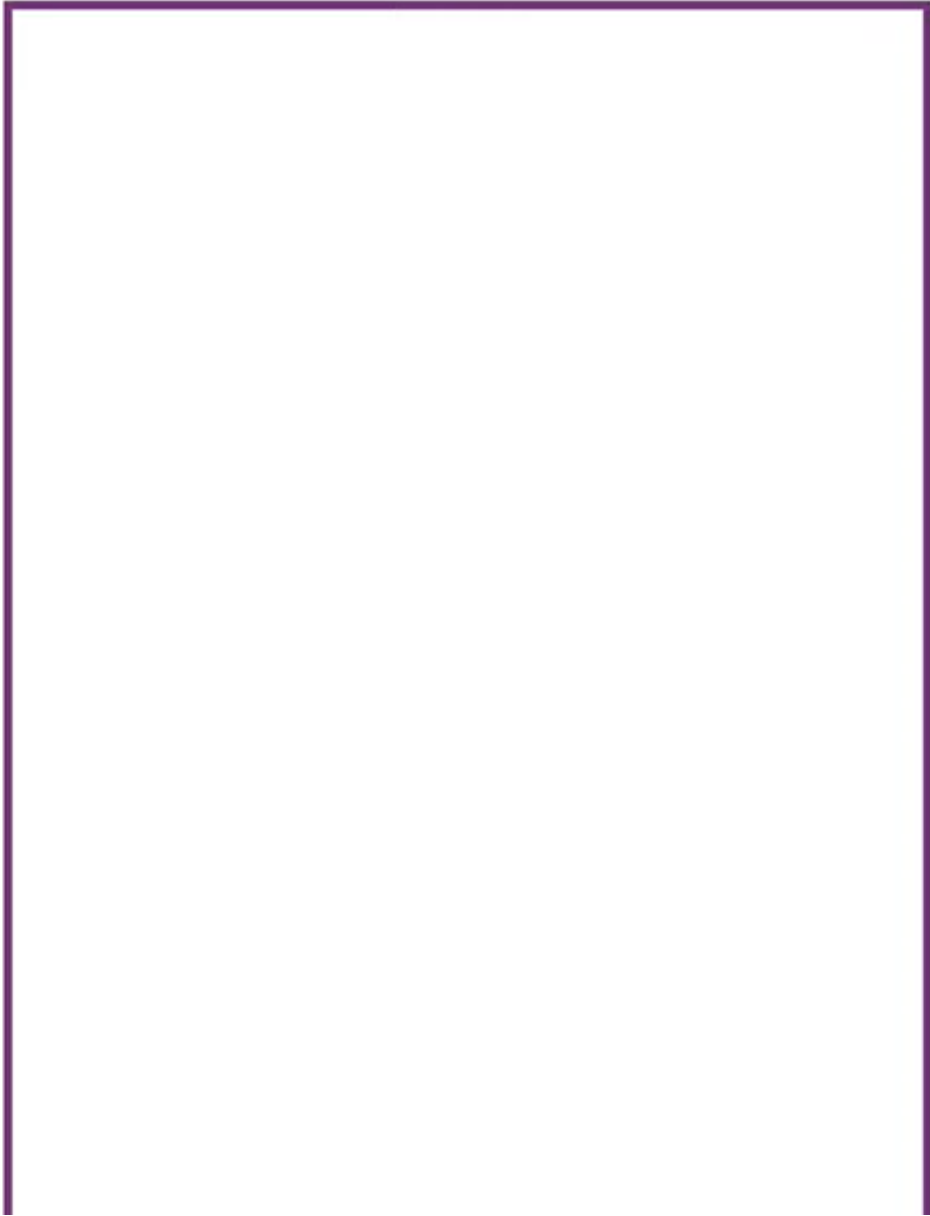
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }


    public void setAge(int age) {
        this.age = age;
    }

    public boolean isElectricVehicle() {
```



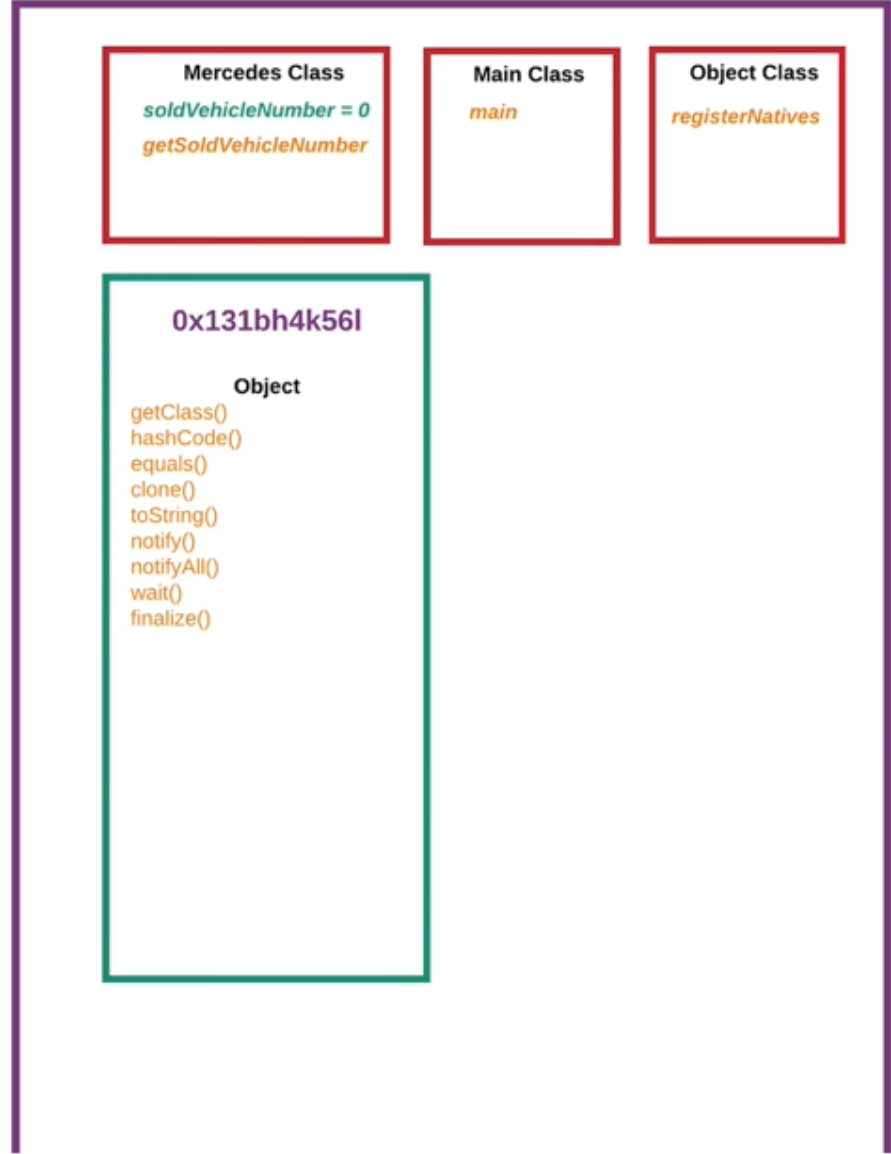
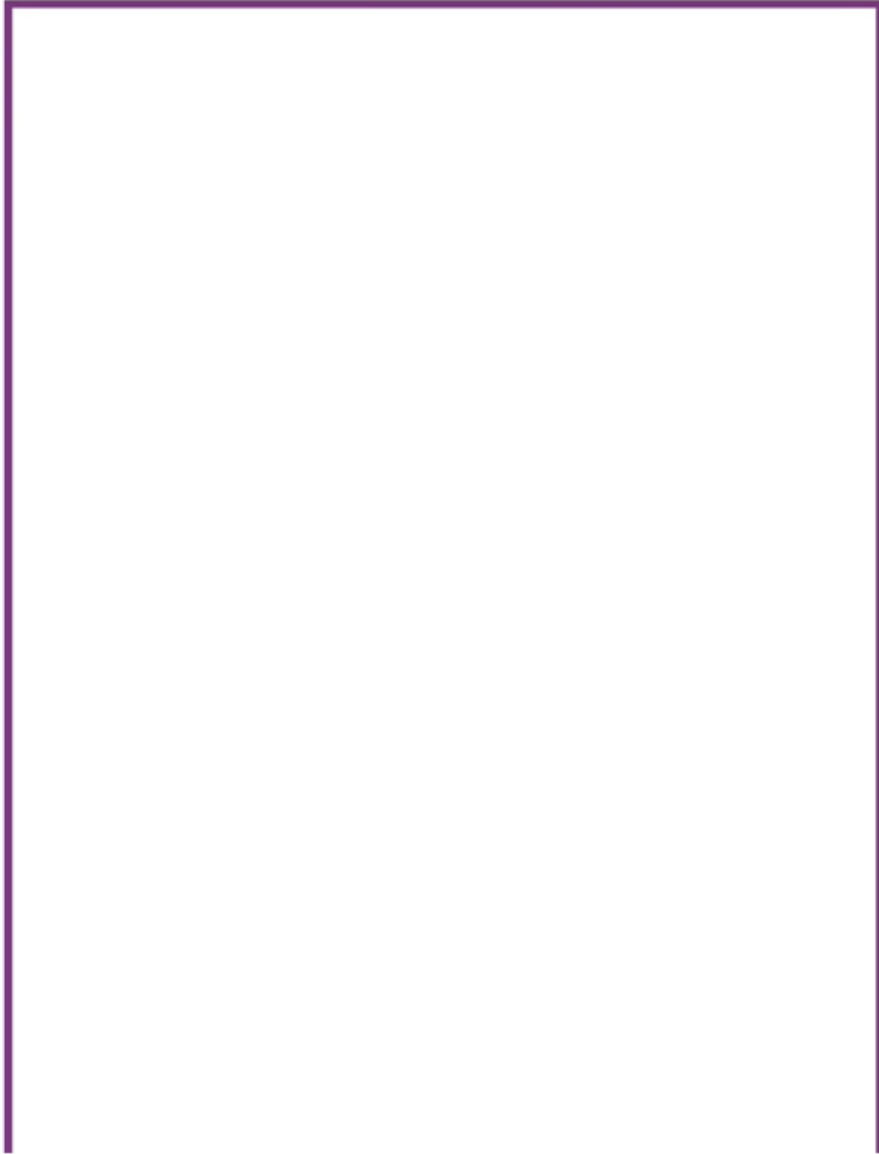
11-new Constructor

```
public class Main {  
    public static void main(String[] args) {  
        Mercedes m1 = new Mercedes( name: "CLA", electrical: true , age: 2);  
        Mercedes m2 = new Mercedes( name: "E200", electrical: false , age: 5);  
        System.out.println("Name: " + m1.getName() + " Age: " + m1.getAge() + " Electrical: " + m1.isElectricVehicle());  
        System.out.println("Name: " + m2.getName() + " Age: " + m2.getAge() + " Electrical: " + m2.isElectricVehicle());  
        System.out.println(Mercedes.getSoldVehicleNumber());  
    }  
}
```



```
public class Mercedes {  
  
    private static int soldVehicleNumber;  
  
    public static int getSoldVehicleNumber() { return soldVehicleNumber; }  
  
    private String name;  
    private BigDecimal serialNumber;  
    private int age;  
    private boolean isElectricVehicle;  
  
    public Mercedes(String name, boolean electrical, int age) {  
        super();  
        this.name = name;  
        this.serialNumber = BigDecimal.valueOf(new Random().nextInt( bound: 1000));  
        this.age = age;  
        soldVehicleNumber++;  
        isElectricVehicle = electrical;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getAge() {  
        return age;  
    }  
}
```





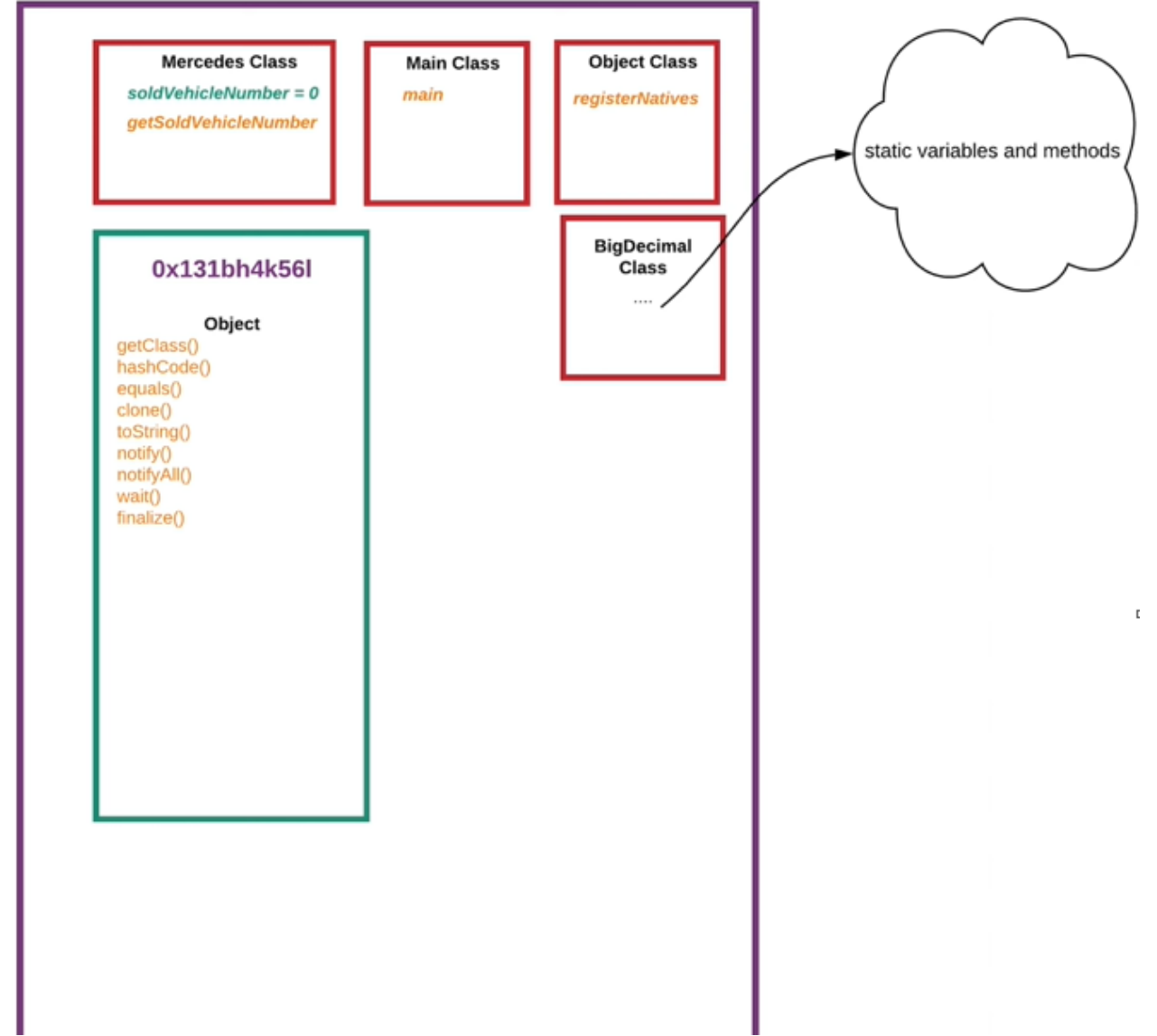
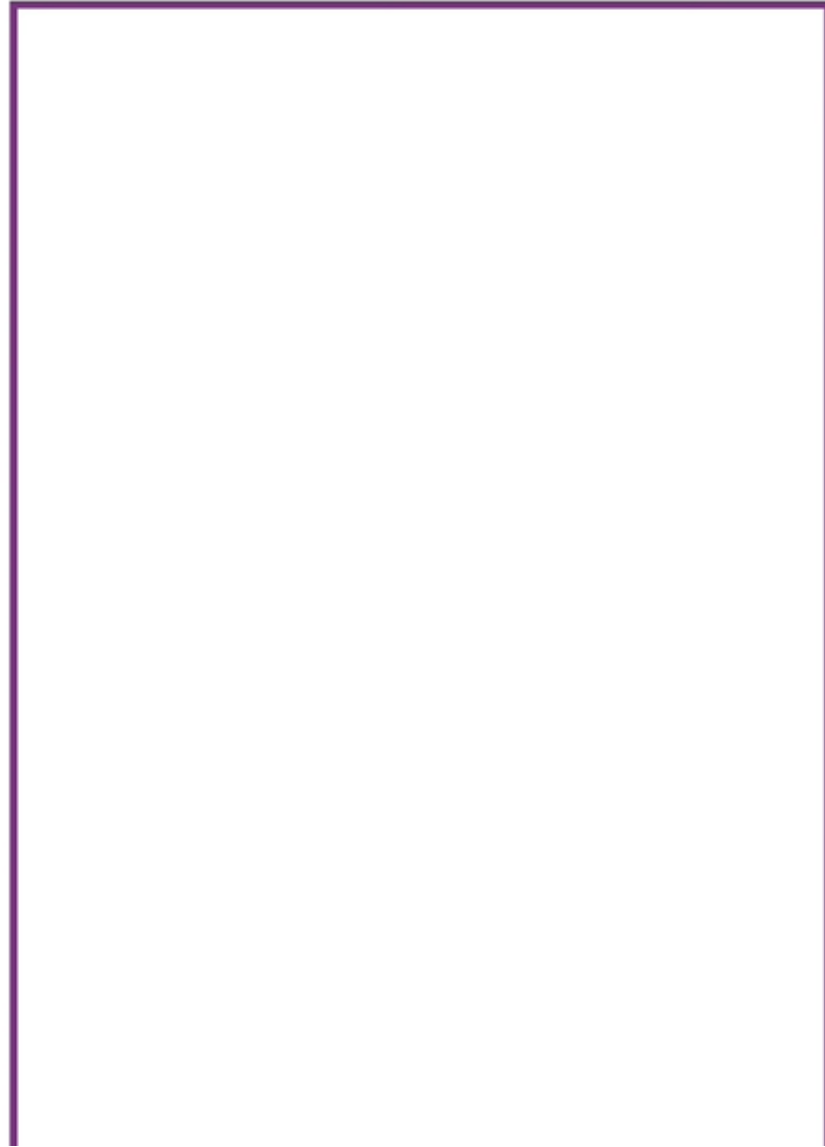

```
} public static int getSoldVehicleNumber() { return soldVehicleNumber; }

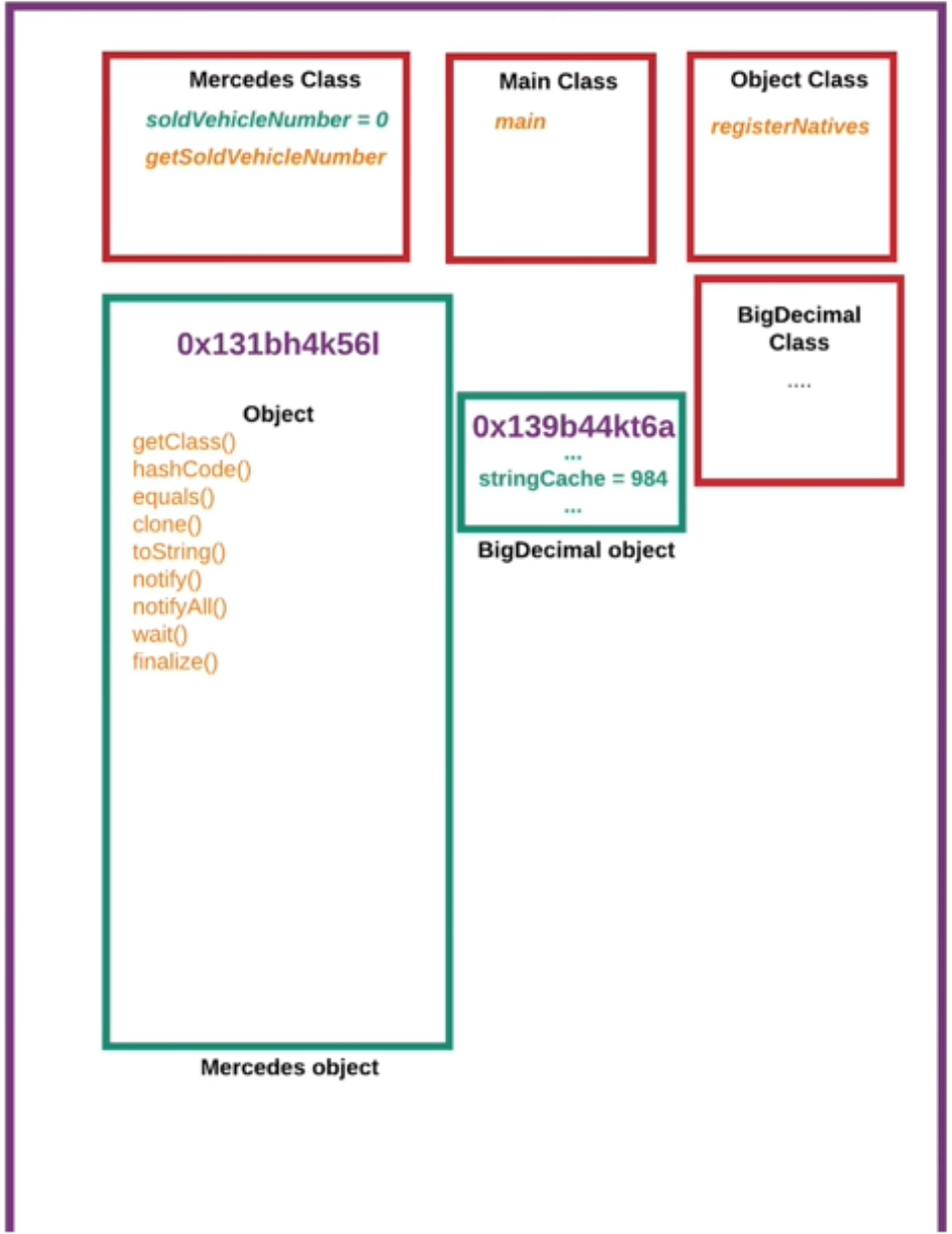
private String name;
private BigDecimal serialNumber;
private int age;
private boolean isElectricVehicle;

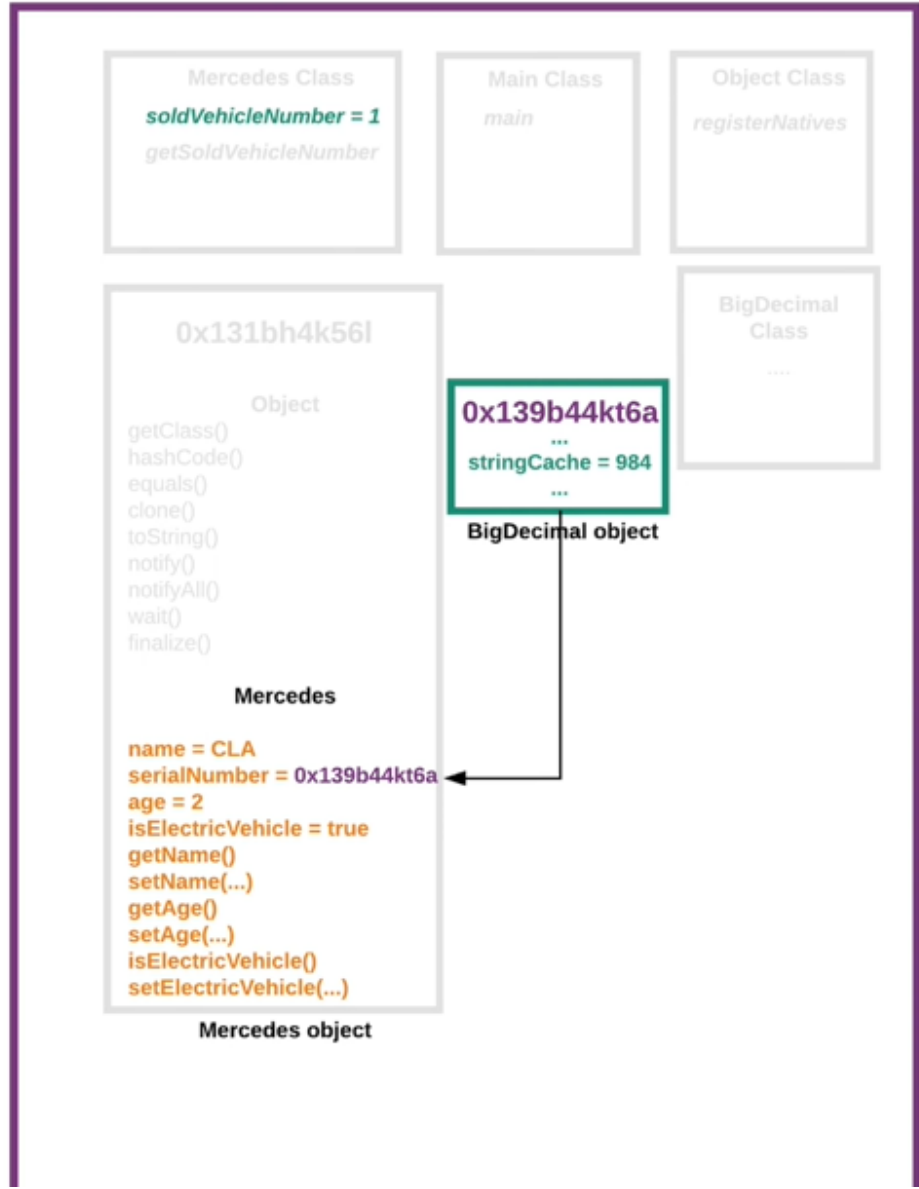
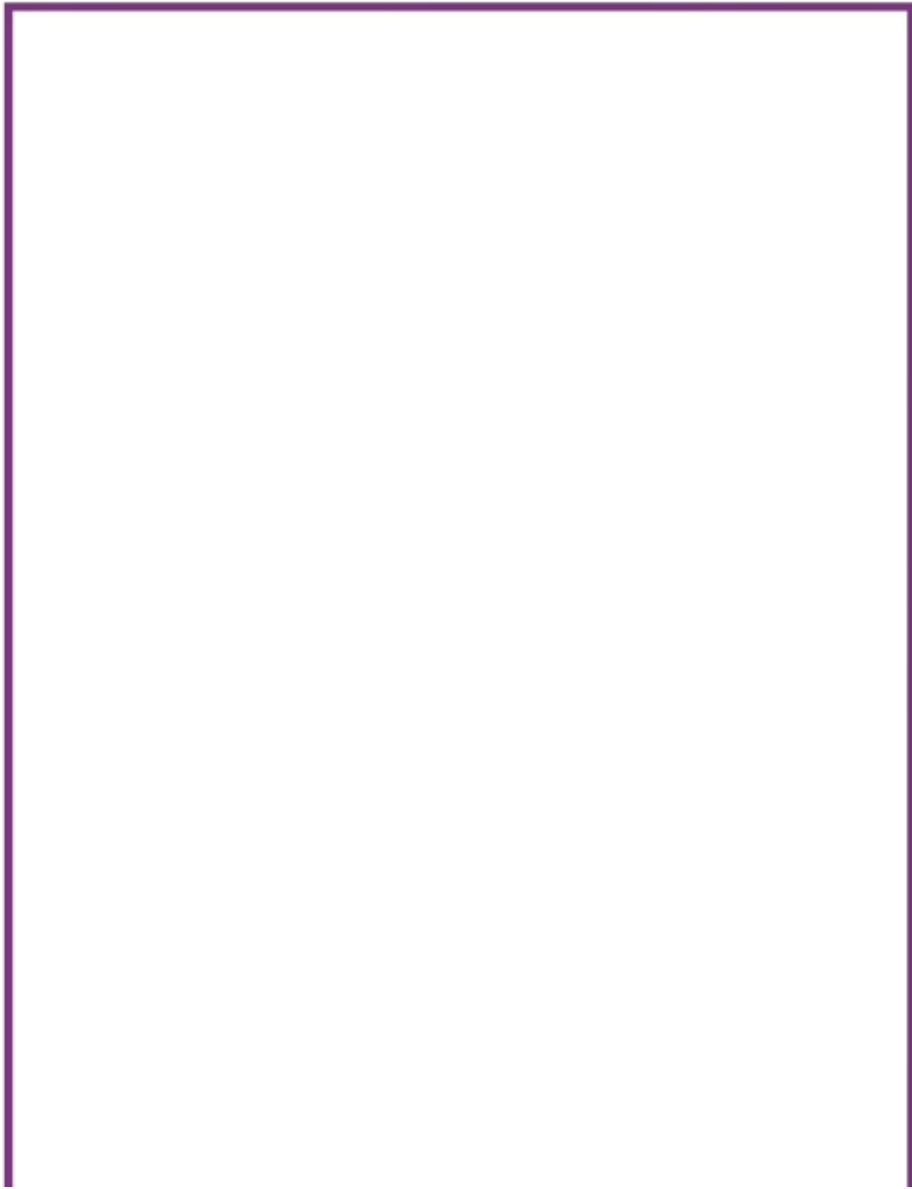
} public Mercedes(String name, boolean electrical, int age) {
    super();
    this.name = name;
    this.serialNumber = BigDecimal.valueOf(new Random().nextInt( bound: 1000));
    this.age = age;
    soldVehicleNumber++;
    isElectricVehicle = electrical;
}

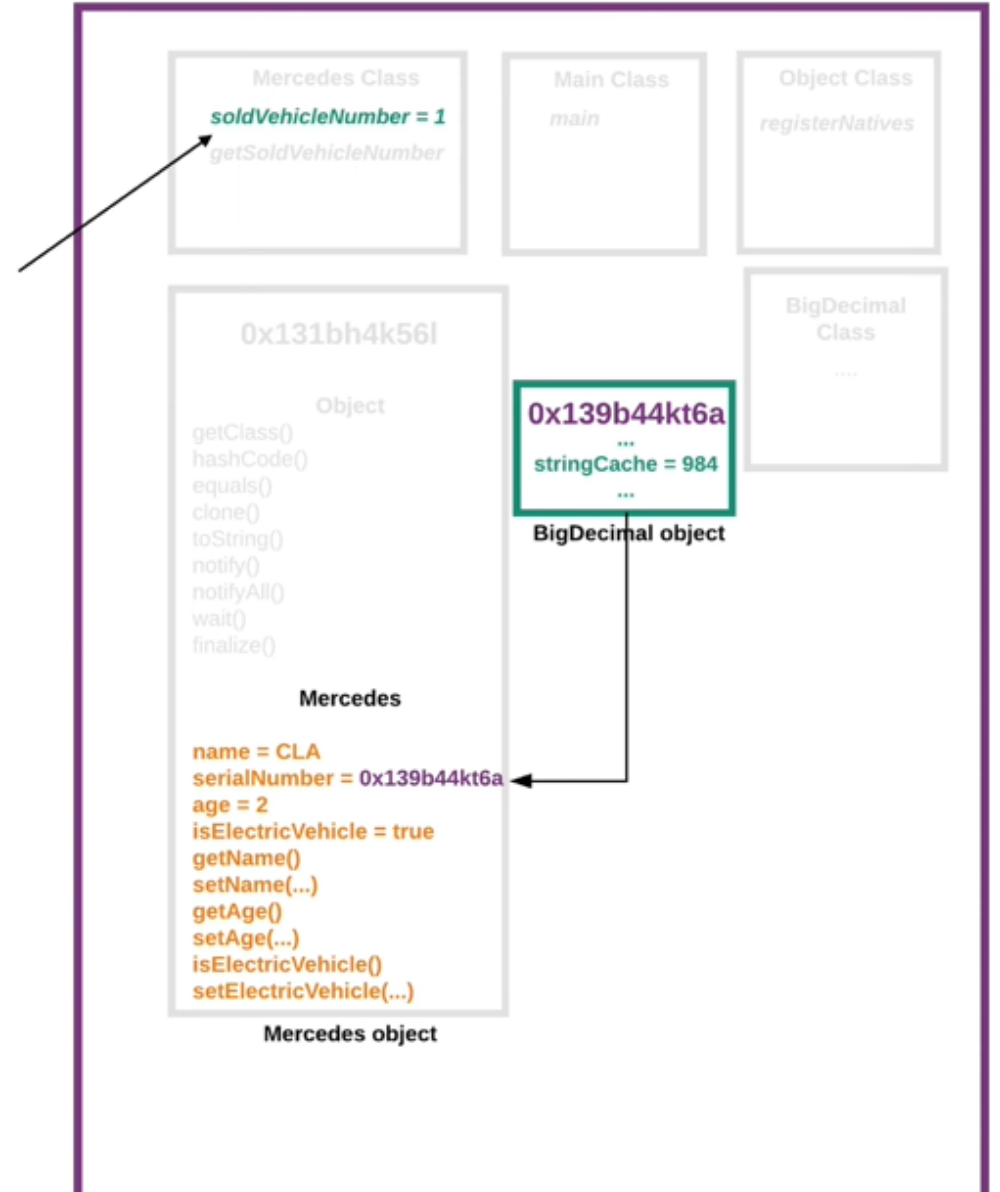
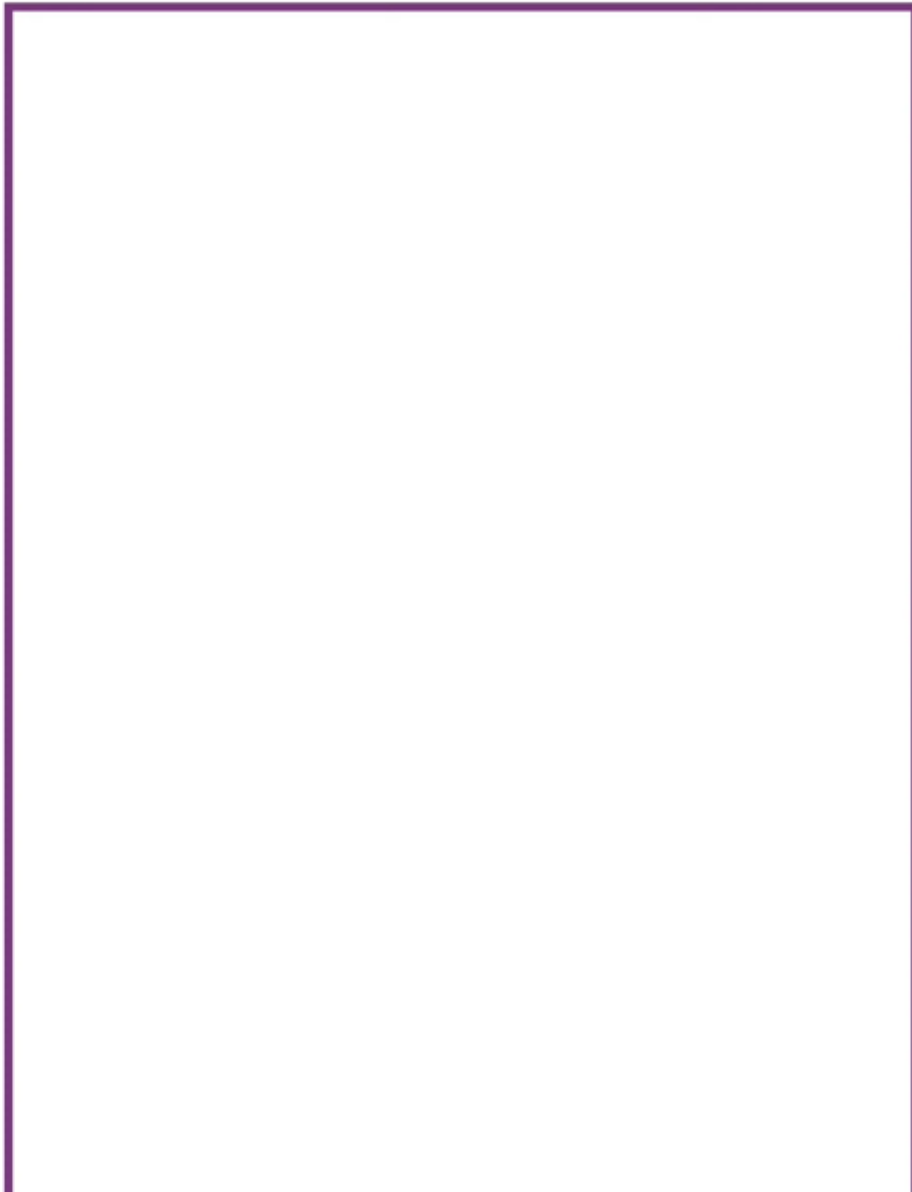
} public String getName() {
    return name;
}

}
```



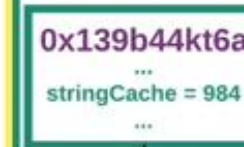
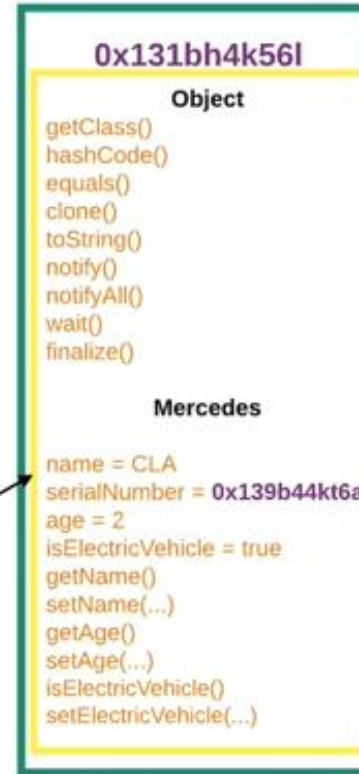
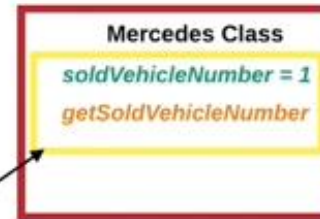






Static context

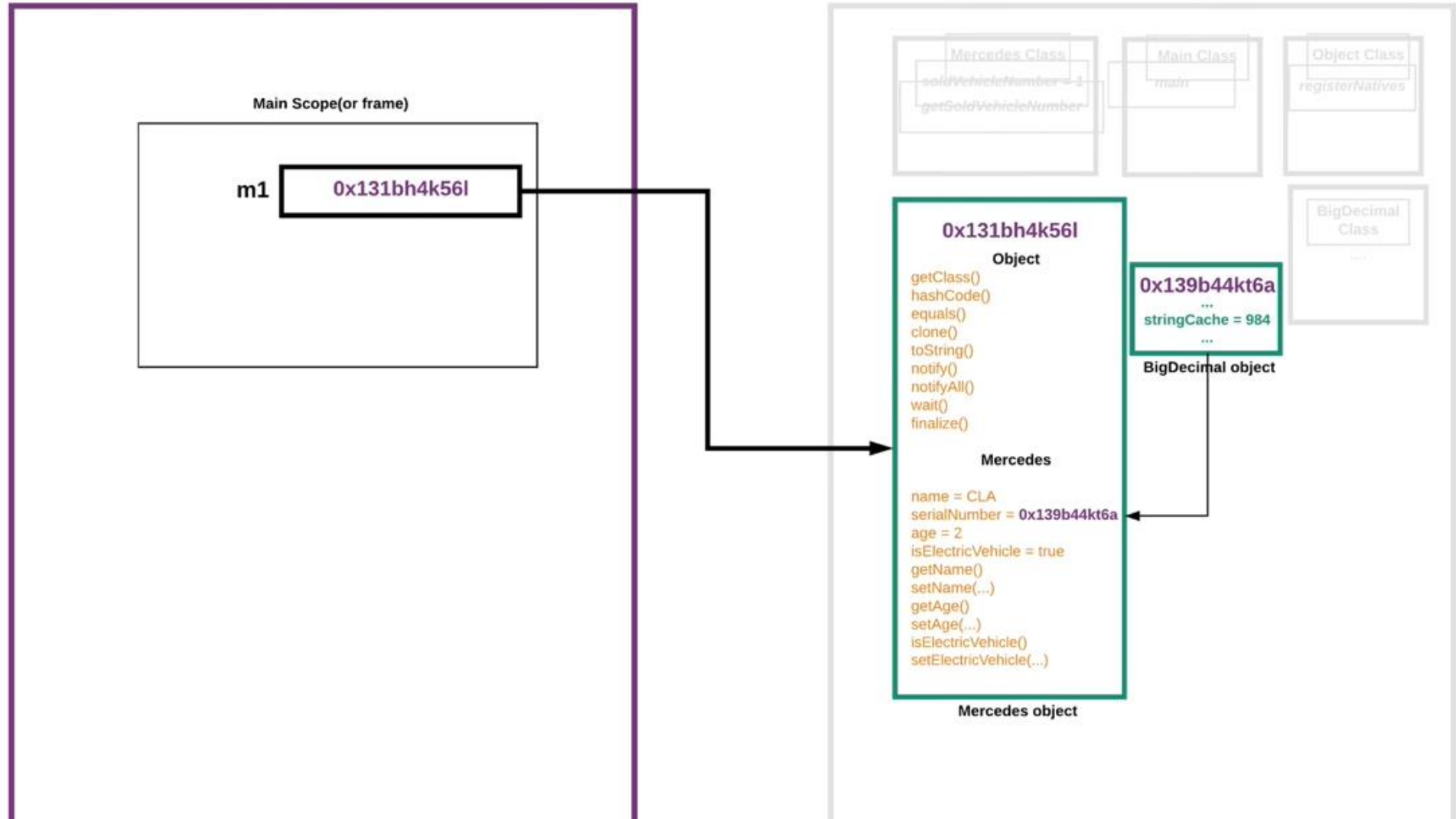
Instance context



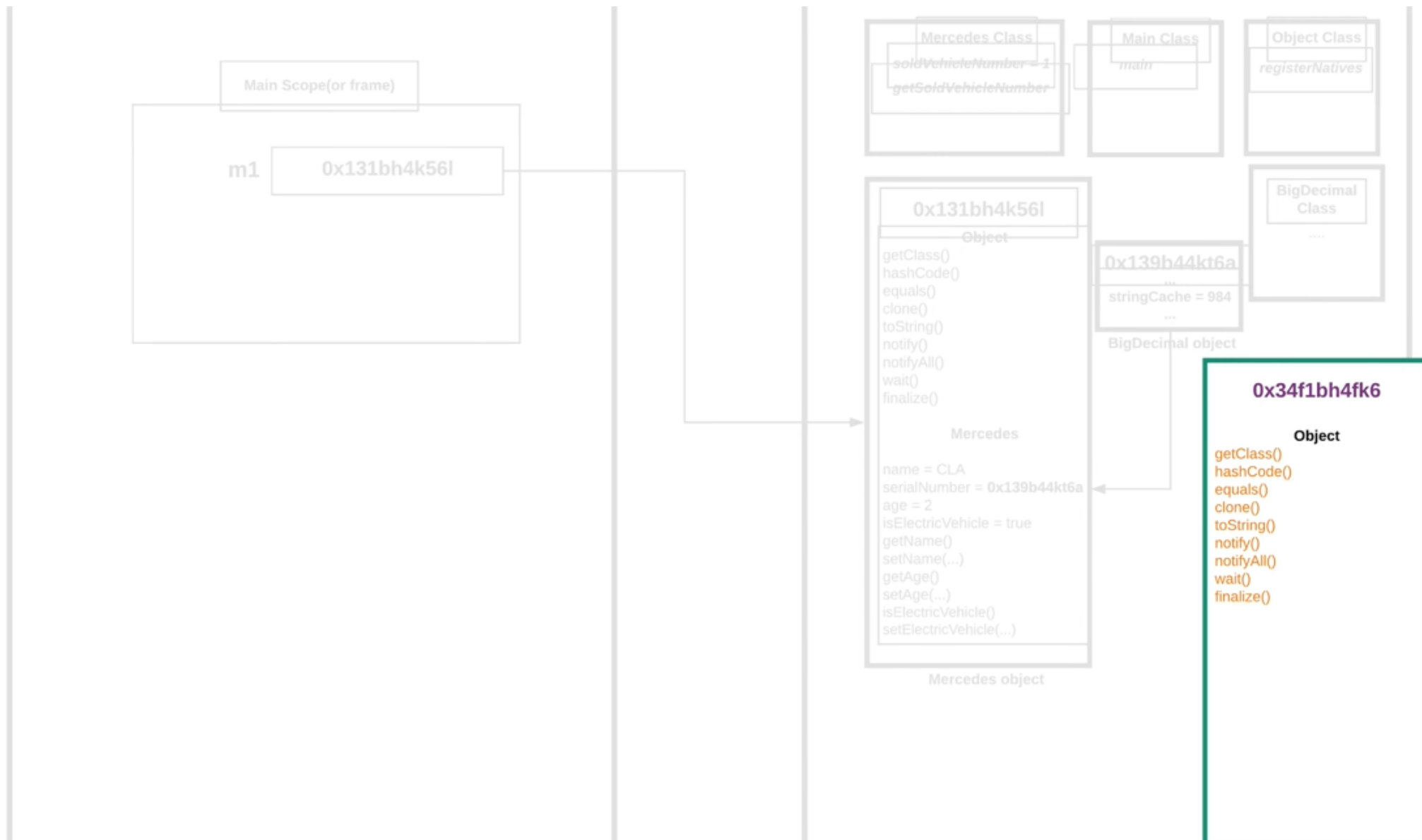
BigDecimal object

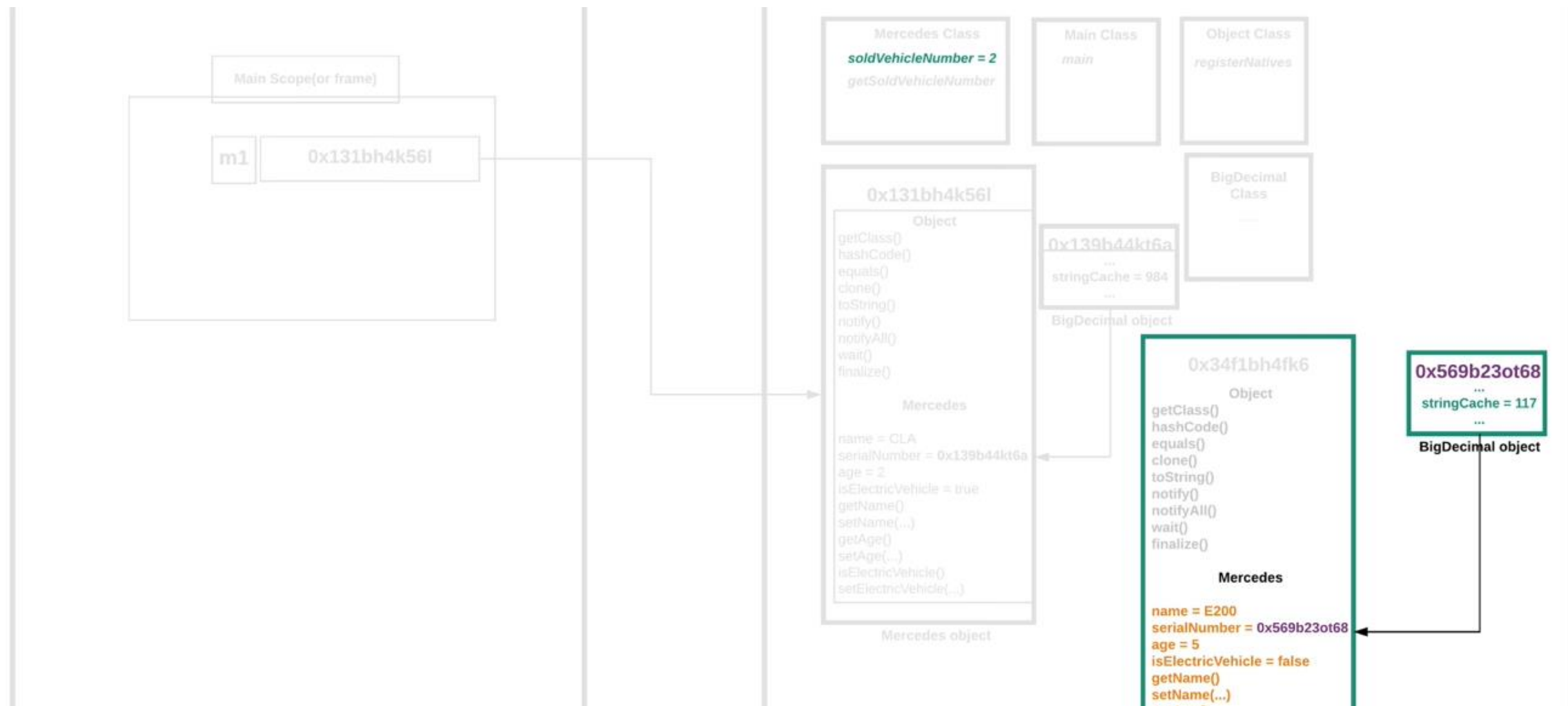
Mercedes object

```
public class Main {  
    public static void main(String[] args) {  
        Mercedes m1 = new Mercedes( name: "CLA", electrical: true , age: 2);  
        Mercedes m2 = new Mercedes( name: "E200", electrical: false , age: 5);  
        System.out.println("Name: " + m1.getName() + " Age: " + m1.getAge() + " Electrical: " + m1.isElectricVehicle());  
        System.out.println("Name: " + m2.getName() + " Age: " + m2.getAge() + " Electrical: " + m2.isElectricVehicle());  
        System.out.println(Mercedes.getSoldVehicleNumber());  
    }  
}
```

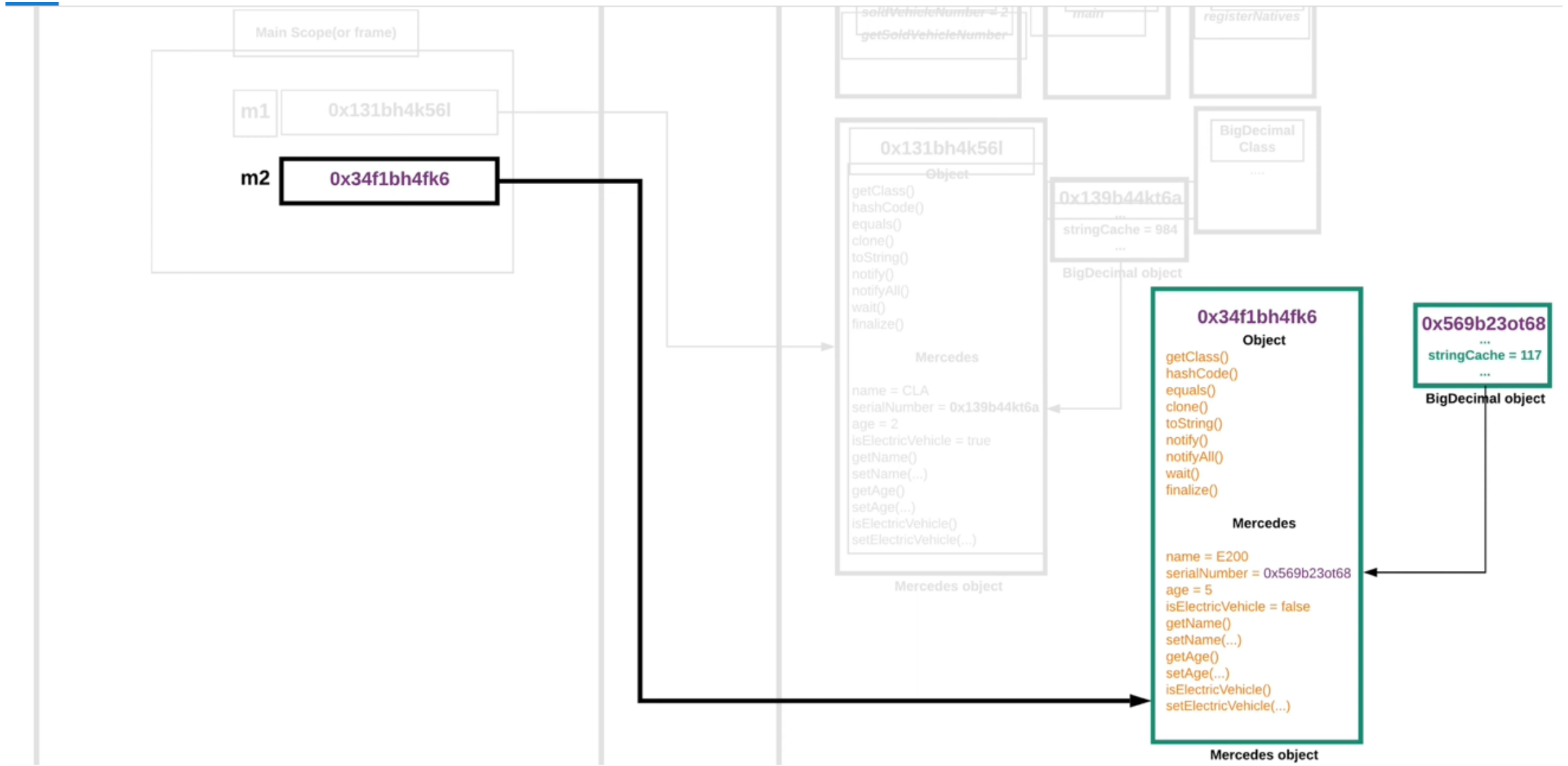


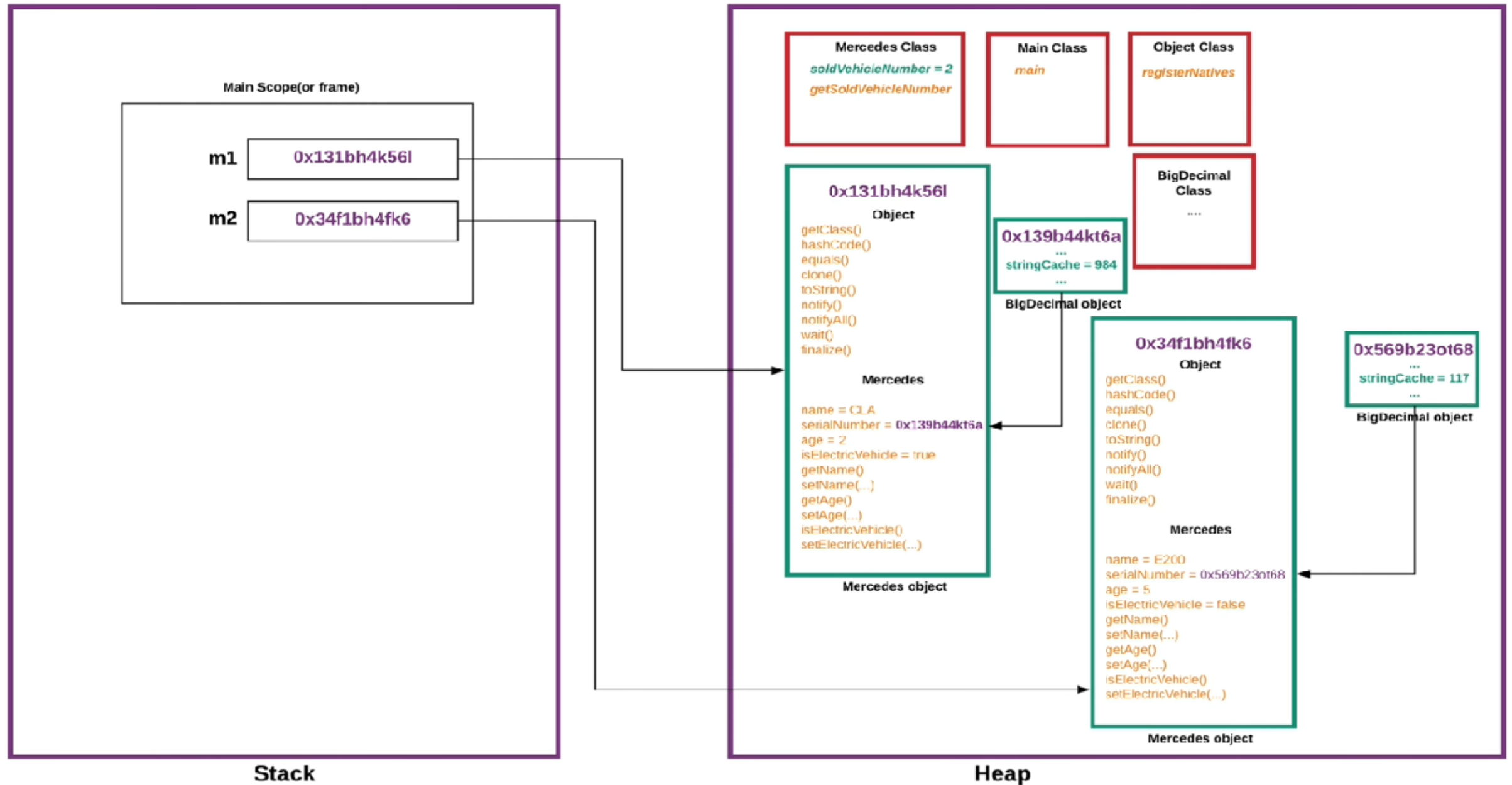

```
public class Main {  
    public static void main(String[] args) {  
        Mercedes m1 = new Mercedes(name: "CLA", electrical: true , age: 2);  
        Mercedes m2 = new Mercedes(name: "E200", electrical: false , age: 5);  
        System.out.println("Name: " + m1.getName() + " Age: " + m1.getAge() + " Electrical: " + m1.isElectricVehicle());  
        System.out.println("Name: " + m2.getName() + " Age: " + m2.getAge() + " Electrical: " + m2.isElectricVehicle());  
        System.out.println(Mercedes.getSoldVehicleNumber());  
    }  
}
```



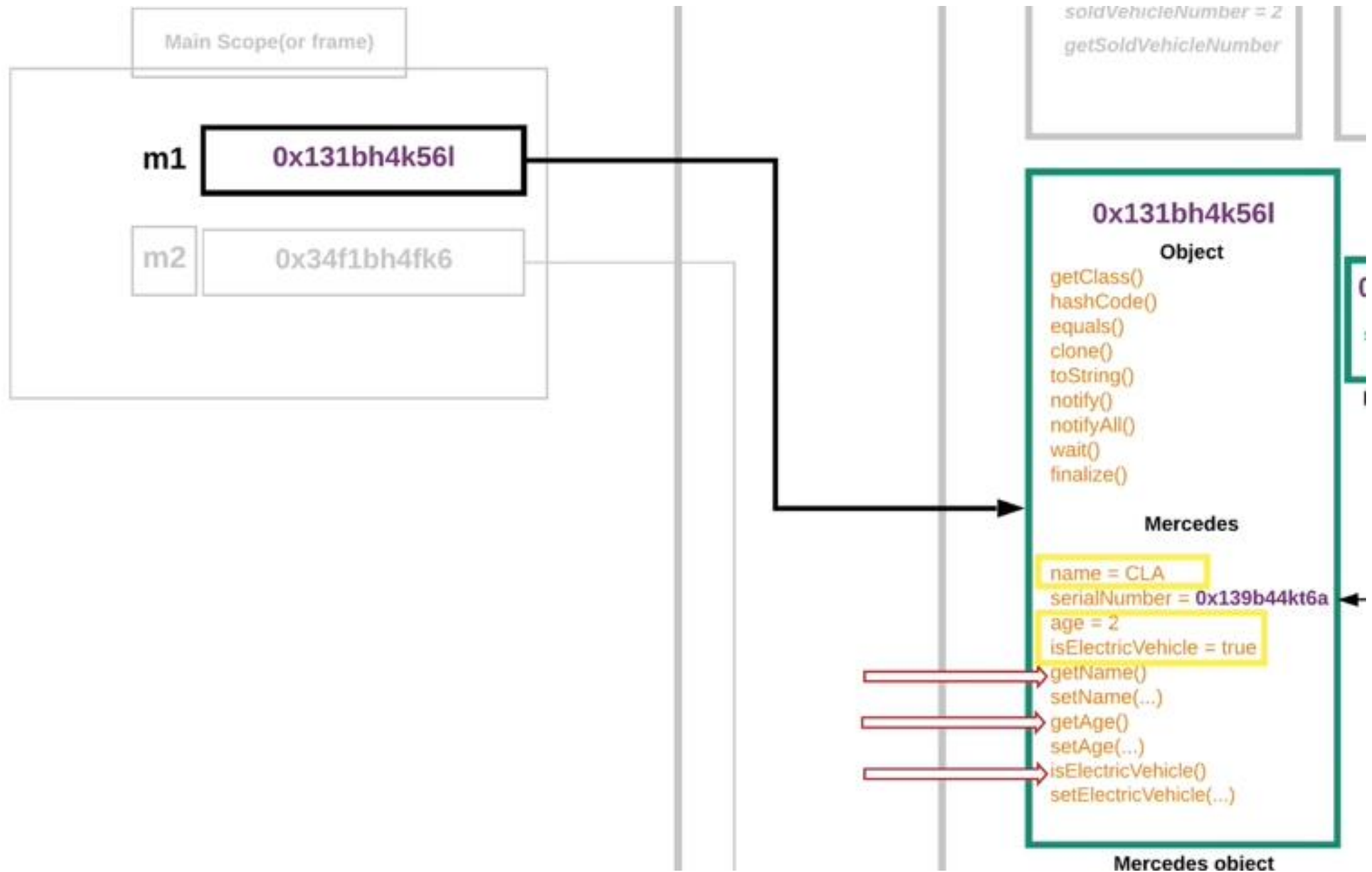


```
public class Main {  
    public static void main(String[] args) {  
        Mercedes m1 = new Mercedes( name: "CLA", electrical: true , age: 2);  
        Mercedes m2 = new Mercedes( name: "E200", electrical: false , age: 5);  
        System.out.println("Name: " + m1.getName() + " Age: " + m1.getAge() + " Electrical: " + m1.isElectricVehicle());  
        System.out.println("Name: " + m2.getName() + " Age: " + m2.getAge() + " Electrical: " + m2.isElectricVehicle());  
        System.out.println(Mercedes.getSoldVehicleNumber());  
    }  
}
```





```
public class Main {  
    public static void main(String[] args) {  
        Mercedes m1 = new Mercedes( name: "CLA", electrical: true , age: 2);  
        Mercedes m2 = new Mercedes( name: "E200", electrical: false , age: 5);  
        System.out.println("Name: " + m1.getName() + " Age: " + m1.getAge() + " Electrical: " + m1.isElectricVehicle());  
        System.out.println("Name: " + m2.getName() + " Age: " + m2.getAge() + " Electrical: " + m2.isElectricVehicle());  
        System.out.println(Mercedes.getSoldVehicleNumber());  
    }  
}
```



Statik Metotlar ve Değişkenler

Statik

- Java nesne yönelimlidir fakat bir özel durumumuz vardır bir yardımcı yöntem bu yöntemde sınıfın bir örneğine sahip olmamıza gerek yoktur.
- Statik anahtar kelimesi bize şunu sağlar bir yöntem veya değişken sınıfın herhangi bir örneği olmadan da çalışır.

```
public class StatikMethods {  
    public static void main(String[] args) {  
        double pi = Math.PI;  
    }  
}
```

- Statik yöntemler , statik yöntemin sınıfının belirli bir örneğini bilmeden çalışır statik yöntem örnek başvurusu direkt Math.Random() gibi çağırıldığından static yöntem sınıfın herhangi bir örnek değişkenine(non-static) başvuramaz. Statik yöntem hangi örneğin değişken değerinin kullanılacağını bilemez.

```
class Duck {  
    4 usages  
    private int size;  
  
    public void setSize(int s) {  
        s = size;  
    }  
  
    public int getSize() {  
        return size;  
    }  
  
    public static void main(String[] args) {  
        System.out.println(size);  
        //size burada statik değil eğer biz bunu çağırmak istiyorsak  
        // Aşağıdaki gibi Duck nesnesi oluşturup o nesnenin referansından çağırabiliriz.  
        Duck d1 = new Duck();  
        System.out.println(d1.size);  
    }  
}
```

STATİK YÖNTEMLER STATİK OLMAYAN YÖNTEMLERİ KULLANAMAZLAR

Statik olmayan yöntemler genellikle yöntemin davranışını etkilemek için örnek değişken durumunu kullanırlar getName(), name değişkeninin değerini döndürür.

```
class Duck {  
    2 usages  
    private String name;  
  
    public void setName(String s)  
    {  
        name = s;  
    }  
    1 usage  
    public String getName()  
    {  
        return name;  
    }  
  
    public static void main(String[] args)  
    {  
        System.out.println("Name is " + getName()); // Yine aynı problem hangi nesnenin getName() metodu ?  
        Duck d1 = new Duck();  
        System.out.println(d1.getName());  
    }  
}
```

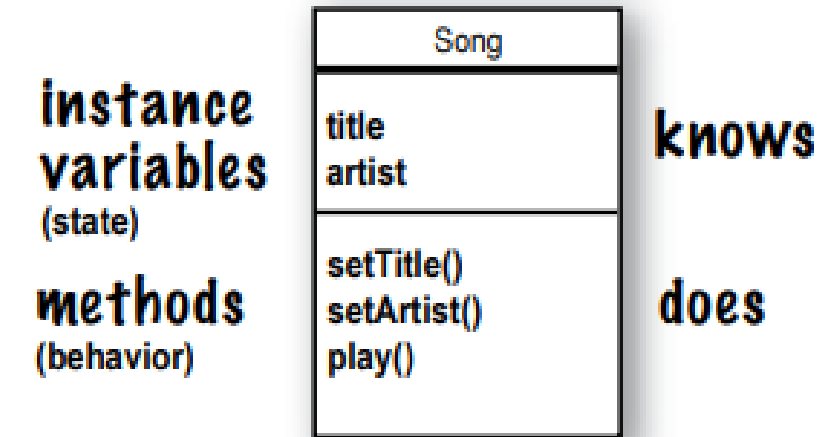
- Statik değişkenlerde o değişkenin değeri tüm sınıflarda aynıdır
- Statik değişkenler paylaşılır. Aynı sınıfın tüm örnekleri statik değişkenlerin tek bir kopyasını paylaşır
- Bir sınıftaki statik değişkenler, o sınıfın herhangi bir nesnesi oluşturulmadan önce başlatılır
- Bir sınıftaki statik değişkenler sınıfın herhangi bir statik statik yöntemi çalışmadan önce başlatılır.

```
class Duck{  
  
    private int size ;  
    1 usage  
    private static int duckCount ; //Statik değişkeni bu classın her örneği için aynı kalır.  
  
    public static void main()  
    {  
        Duck dk = new Duck(); //Bir duck nesnesi duckCountın bir kopyasını tutamaz.  
        // duckCount statik olduğundan duck nesnelerinin tümü tek bir kopyasını paylaşır  
  
        Duck dk1 = new Duck();  
  
        System.out.println(duckCount);  
    }  
}
```

```
//Mesela aynı nesneden toplam kaçtane üretmişiz onu sayacak bir kod örneği yazalım  
class Duck{  
    1 usage  
    public int count;  
    //Böyle olduğunda saymaz çünkü nesne her oluştuğunda count da onunla birlikte yeniden oluşur ve sıfırlanır  
  
    public Duck(){  
        count++;  
    }  
}
```

Instance Variables

- Biz bir sınıf oluştururken şunlara dikkat ederiz
- Bir nesnenin bildiği şeyler, örnek değişkenleridir(Instance Variables)



```
public class StatikMethods{
    public static void main(String[] args) {
        Duck d1 = new Duck();

        System.out.println(d1.size);
        System.out.println(d1.point);
        System.out.println(d1.bool);
        System.out.println(d1.str);
    }
}

2 usages
class Duck{
    public int size ;
    1 usage
    public float point;
    1 usage
    public boolean bool;
    public String str ;
}

StatikMethods x
"C:\Program Files\Java\jdk-17.0.4\bin\java.exe" "-javaagent:
0
0.0
false
null

Process finished with exit code 0
```

-
- Örnek değişkenleri her zaman default bir değer alır.
(int = 0 , float = 0.0 , boolean = false, references = null)
 - Örnek değişkenleri bir sınıf içinde bildirilir, ancak bir yöntem içinde bildirilmez.
 - Örnek değişkenleri, ait oldukları nesnenin içinde yaşar.
 - Yığındaki bir nesne için alan ayrıldığında, her örnek değişkeni değeri için bir alan oluşturulur.

