JAVA PROGRAMMING BASICS

Module 2: Java Object-oriented Programming

Training program

1. Classes and Instances
2. The Methods
3. The Constructors
4. Static elements
5. Initialization sections
6. Package
7. Inheritance and Polymorphism
8. Abstract classes and interfaces
9. String processing
10. Exceptions and Assertions
11. Nested classes
12. Enums
13. Wrapper classes for primitive types
14. Generics
15. Collections
16. Method overload resolution
17. Multithreads
18. Core Java Classes
19. Object Oriented Design

Module contents

- Classes and Instances
  - Overview of class declarations. Class Fields and Methods.
  - Access modifiers
  - Encapsulation
  - Creating Objects
  - null literal
  - this Keyword

Module contents

- Classes and Instances
  - Overview of class declarations. Class Fields and Methods.
  - Access modifiers
  - Encapsulation
  - Creating Objects
  - null literal
  - this Keyword

What Is a Class?

- A class is a blueprint or prototype from which objects are created.
- A class models the state and behavior of a real-world object and class can cleanly model state and behavior.

What Is an Object?

- An object is a software bundle of related state and behavior. Software objects are often used to model the real-world objects.

- An object is a software bundle of variables and related methods.
- Objects are related to real life scenario
- Class is the general thing and object is the specialization of general thing
- Objects are instance of classes.

- An object is characterized by concepts like:

-Attribute

-Behavior

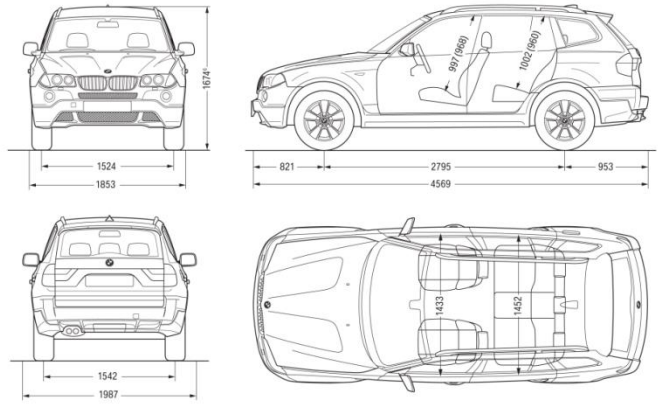
-Identity

Classes and Instances

- "Class" means a category of things
- – A class name can be used as the type of a field or local variable
- "Object" means a particular item that belongs to a class
- – Also called an "instance"

Class declaration

```
public class Car {
    private String model;
    private int maxSpeed;
    private int year;
    //...
    public int getMaxSpeed() {
        return maxSpeed;
    }
    //...
}
```



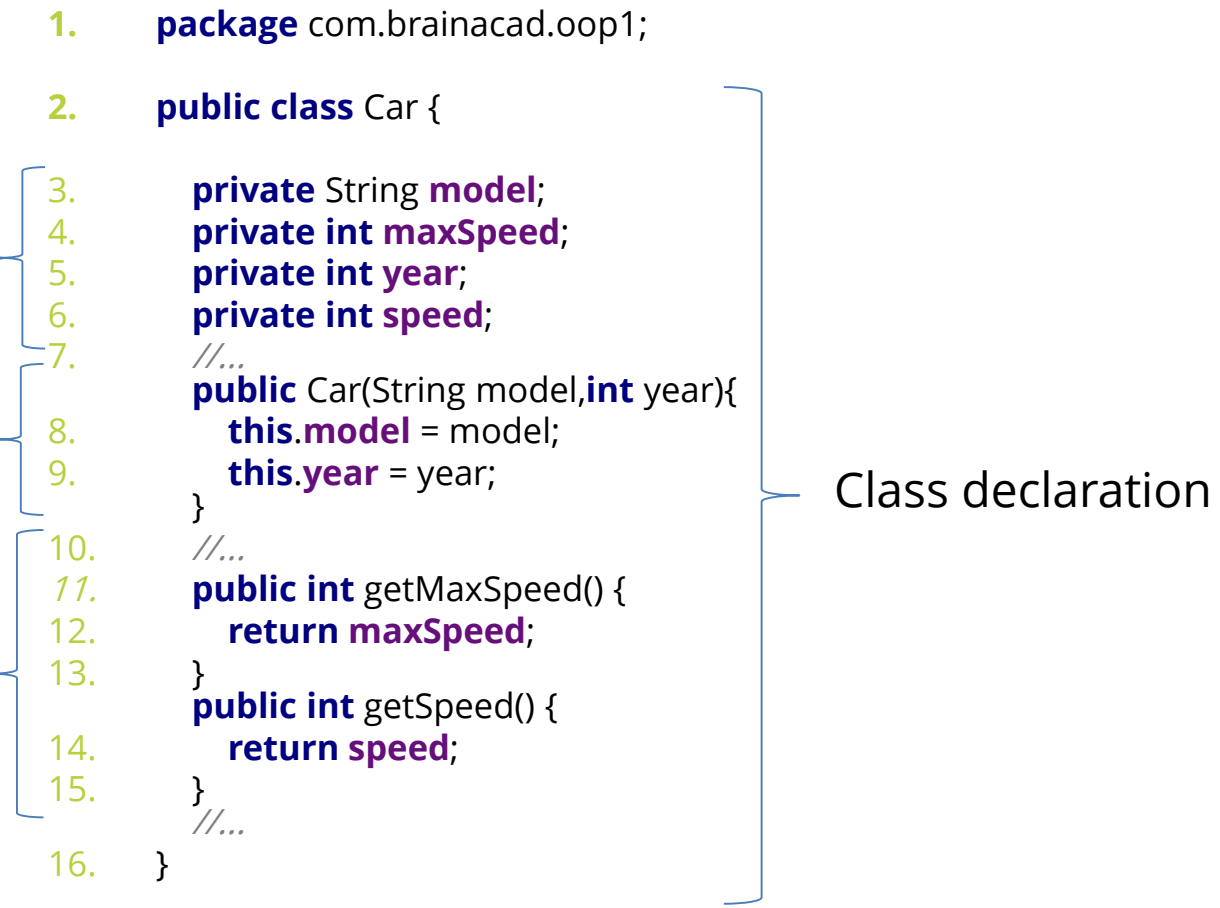
Creating Instances of a Class

Car c1 = new Car(…);
Car c4 = new Car(…);

Class Instances (Objects)

Class Fields and Methods

- Package
- Fields
- Constructor
- Methods



Module contents

- Classes and Instances
  - Overview of class declarations. Class Fields and Methods.
  - Access modifiers
  - Encapsulation
  - Creating Objects
  - null literal
  - this Keyword

Access modifiers

1. public
2. private
3. protected
4. default (not actually an access specifier)

public access modifier

```
1.  public class Car {
2.      //...
3.      public int speed; // public access modifier
4.      //...
5.      public int getSpeed() { // public access modifier
6.          return speed;
7.      }
8.      //...
9.      public void testModifier() {
10.         getSpeed();// access  in  same class  -Ok!
11.     }
12. }
```

default (package) Access Modifier

```
1.  public class Car {
2.      //...
3.      int speed; // default access modifier
4.     //...
5.      int getSpeed() { // default access modifier
6.          return speed;
7.      }
8.      //...
9.      public void testModifier() {
10.         getSpeed();// Ok!
11.     }
12. }
```

protected access modifier

```
1.  public class Car {
2.      //...
3.      protected int speed; // protected access modifier
4.      //...
5.      protected int getSpeed() { // protected access modifier
6.          return speed;
7.      }
        //...
8.      public void testModifier() {
9.          getSpeed();// Ok!
10.     }
11. }
```

private access modifier

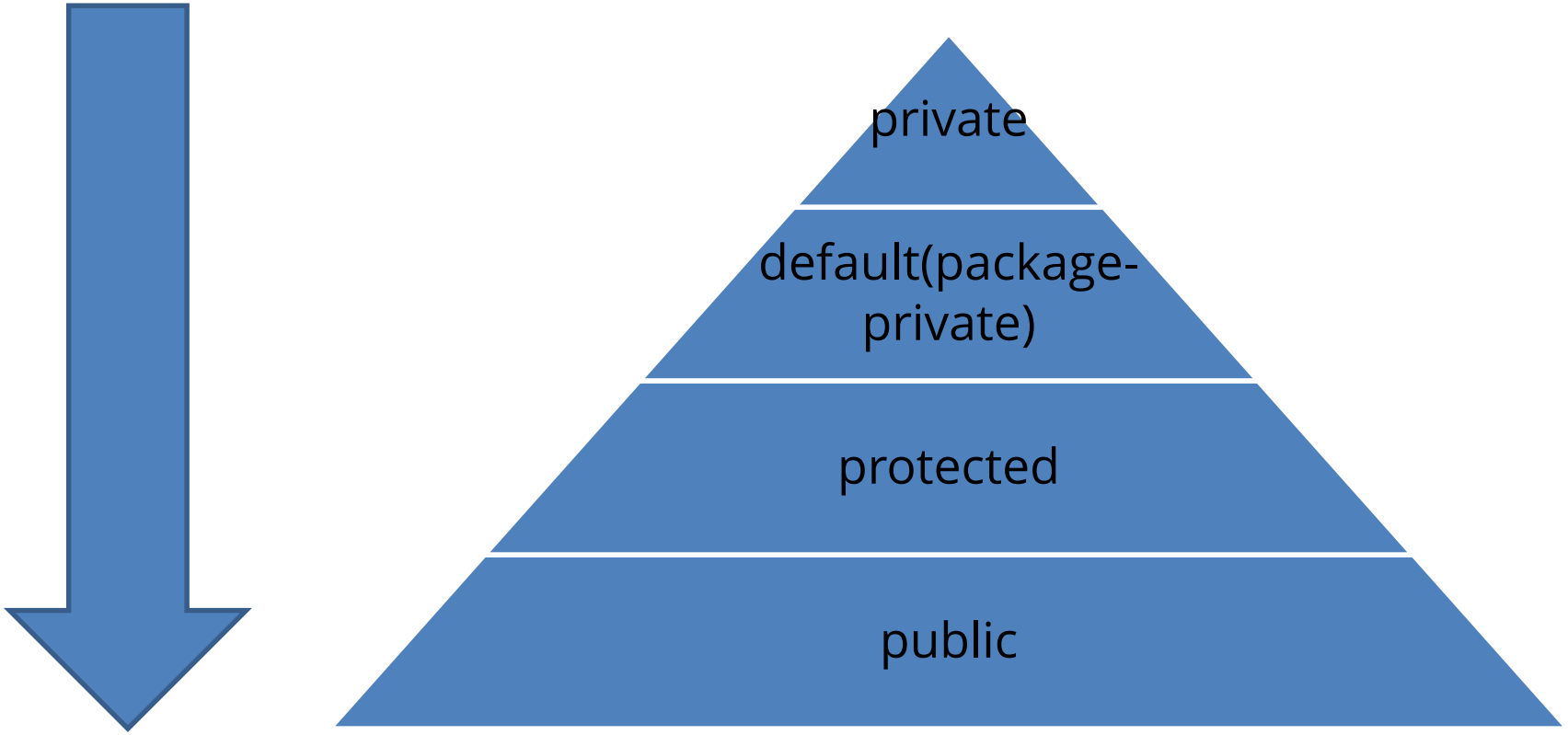
```
1.  public class Car {
2.      //...
3.      private int speed; //  private access modifier
4.      //...
5.      private int getSpeed() { // private access modifier
6.          return speed;
7.      }
        //...
8.      public void testModifier() {
9.          getSpeed();// Ok!
10.     }
11. }
```

Access modifiers 1/2

Visibility	Public	Protected	Default	Private
From the same class	Yes	Yes	Yes	Yes
From any class in the same package	Yes	Yes	Yes	No
From a subclass in the same package	Yes	Yes	Yes	No
From a subclass outside the same package	Yes	Yes, through inheritance	No	No
From any non-subclass outside the package	Yes	No	No	No

Access modifiers 2/2

- **Access Level**



Module contents

- Classes and Instances
  - Overview of class declarations. Class Fields and Methods.
  - Access modifiers
  - Encapsulation
  - Creating Objects
  - null literal
  - this Keyword

Encapsulation 1/3

- The problem:

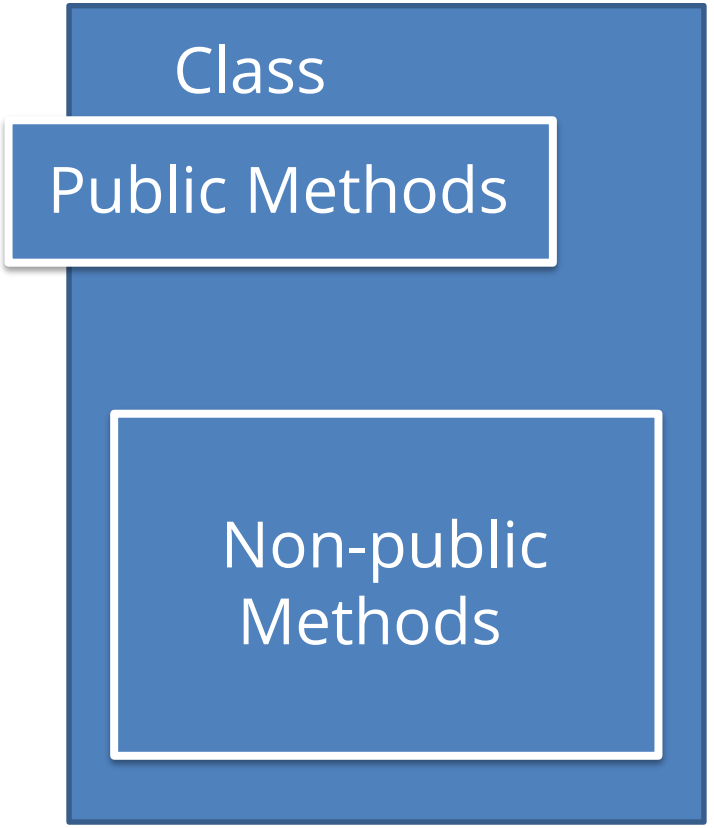
```
1.  public class Car {
2.      //...
3.      // public access modifier
4.      public int speed;
        //...
5.      // public access modifier
6.      public int getSpeed() {
7.          return speed;
8.      }
9.      //...
10. }
```

The problem: value field "speed" can be changed from anywhere outside the class

Encapsulation 2/3

- Encapsulation is one of the four fundamentals of the Object oriented programming.

- What is Encapsulation?
- Encapsulation is a language mechanism to restrict the access of the Objects components to other Objects or Classes.



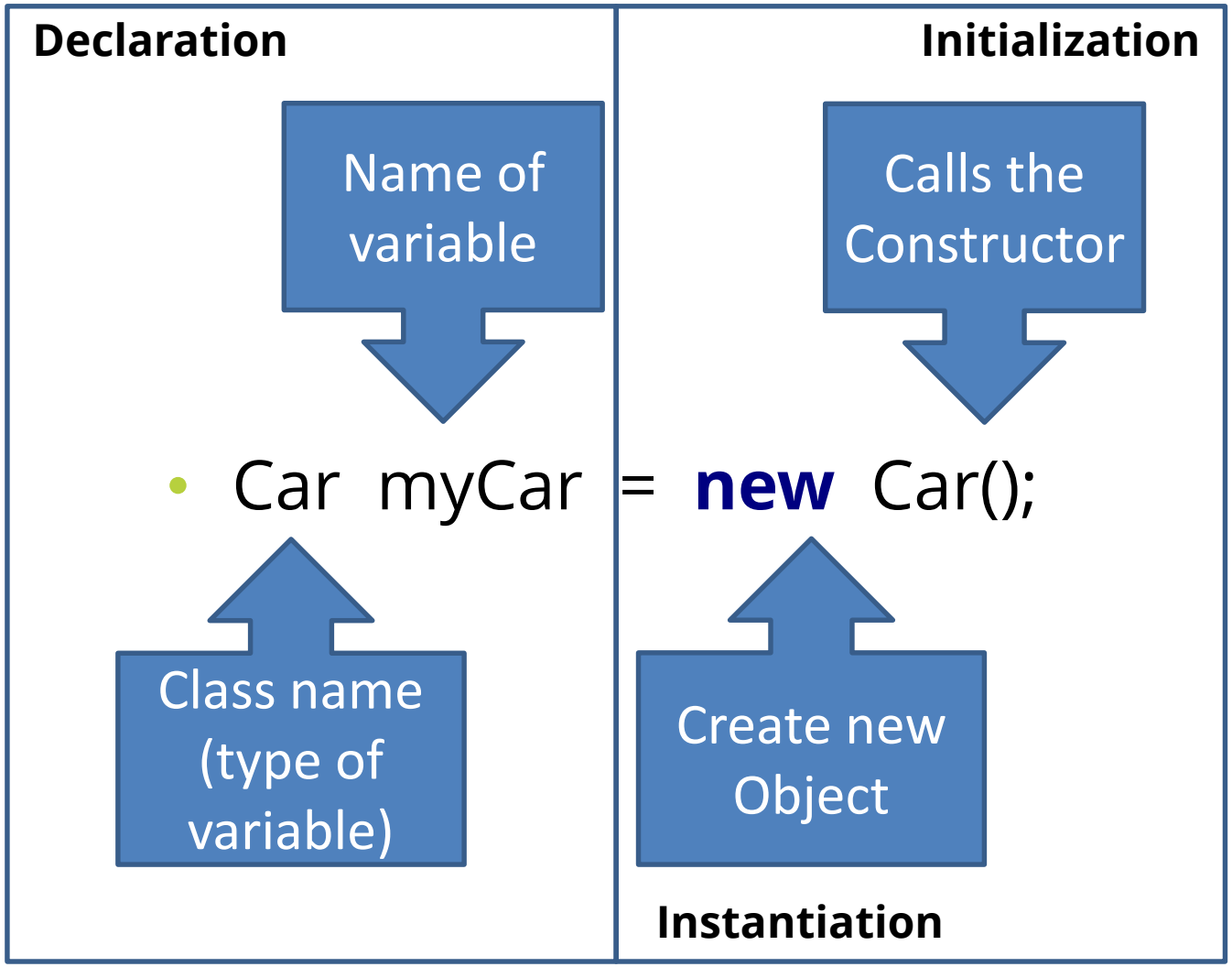
Encapsulation 3/3

```
1.  public class Car {
2.      //...
3.      // private access modifier
4.      private int speed;
5.      //...
6.      // public access modifier
7.      public int getSpeed() {
8.          return speed;
9.      }
10.     //...
11. }
```

Module contents

- Classes and Instances
  - Overview of class declarations. Class Fields and Methods.
  - Access modifiers
  - Encapsulation
  - Creating Objects
  - null literal
  - this Keyword

Creating Objects 1/3

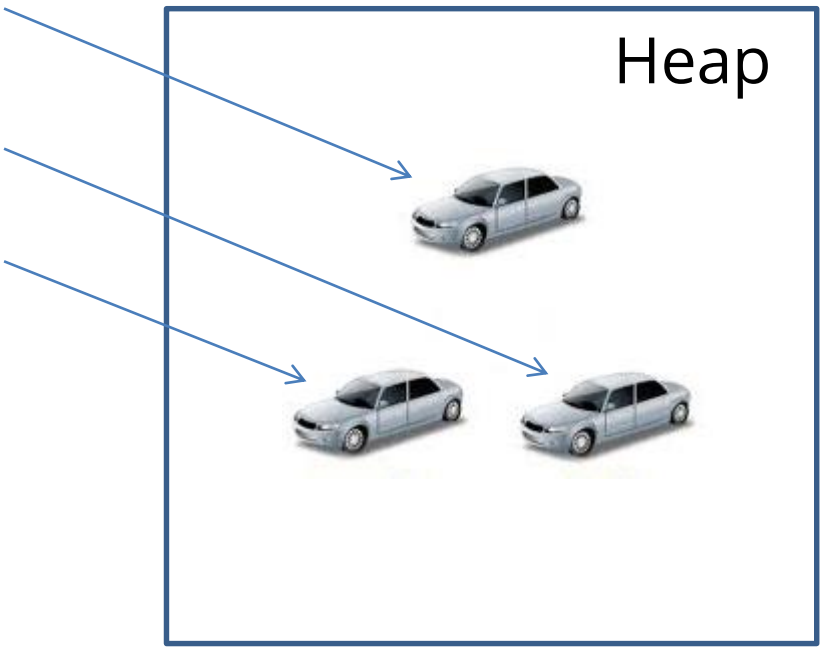


Creating Objects 2/3

```
1. public static void main(String[] arg) {

2.     Car car1 = new Car();
3.     Car car2 = new Car();
4.     Car car3 = new Car();

5. }
```



Creating Objects 3/3

```
1. public static void main(String[] arg) {

2.     Car car1 = new Car();
3.     Car car2 = new Car();
4.     Car car3 = new Car();

5.     // call method "getSpeed" of car1
6.     int speed1 = car1.getSpeed();
7.     // call method "getSpeed" of car2
8.     System.out.println(car2.getSpeed());
9. }
```

Module contents

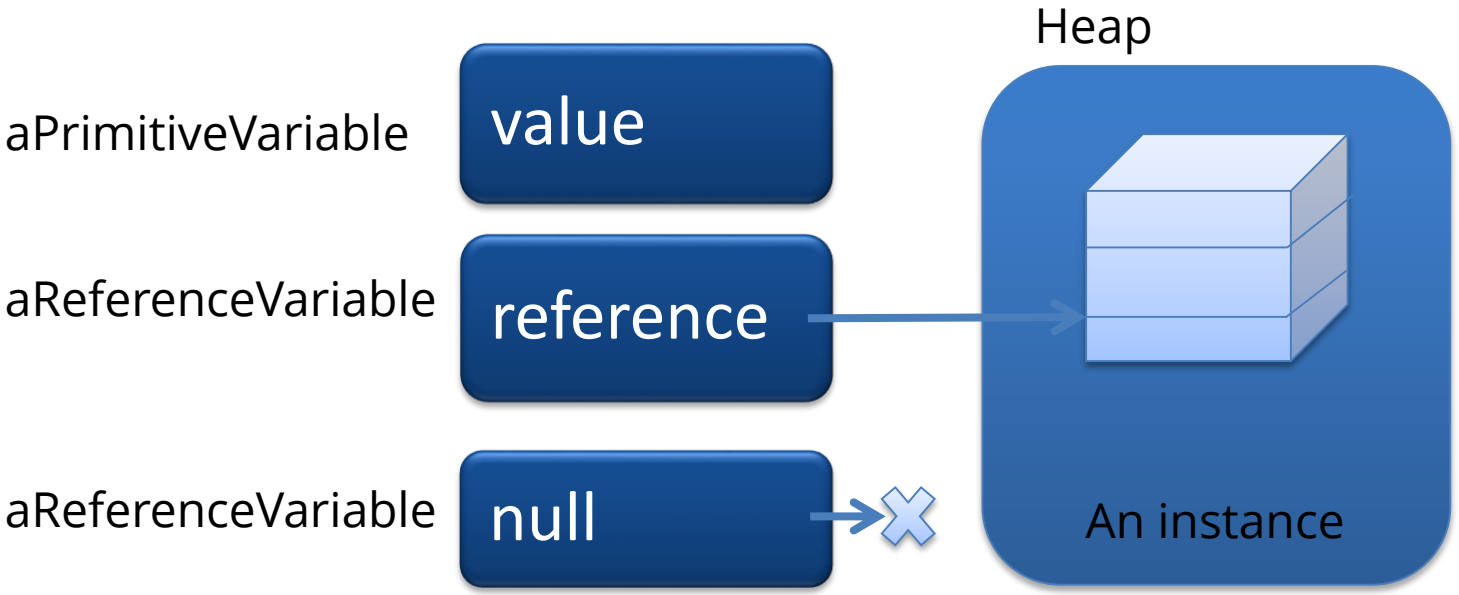
- Classes and Instances
  - Overview of class declarations. Class Fields and Methods.
  - Access modifiers
  - Encapsulation
  - Creating Objects
  - null literal
  - this Keyword

The null 1/3

- A reference variable refers to an object.
- When a reference variable does not have a value (it is not referencing an object) such a reference variable is said to have a null value.
- The null reference can always be assigned or cast to any reference type

The null 2/3

- null indicates that the object reference is not currently referring to an object



The null 3/3

1. String str1 = **null**; *// null can be assigned to String*
2. Car car1 = **null**; *// null can be assigned to Car*
3. **int** i = **null**; *// type mismatch : cannot convert from null to int*

4. String str2 = (String) **null**; *// null can be type cast to String*
5. Car car2 = (Car)**null**; *// null can be type cast to Car*

6. System.***out***.println(**null** == **null**); *// true*
7. System.***out***.println(car1 == **null**); *// true*
8. System.***out***.println(car1 == car2); *// true*

9. car1.getMaxSpeed();

- Exception in thread "main“ java.lang.NullPointerException

Module contents

- Classes and Instances
  - Overview of class declarations. Class Fields and Methods.
  - Access modifiers
  - Encapsulation
  - Creating Objects
  - null literal
  - this Keyword

this keyword 1/2

- **this** is a reference to the *current object* — the object whose method or constructor is being called.

- You can refer to any member of the current object from within an instance method or a constructor by using **this**.

this keyword 2/2

```
public class Car {

    private String model;
    private int maxSpeed;
    private int year;
    private int speed;
    //...

    public void setSpeed(int speed) {
        this.speed = speed;
    }

    //...
}
```