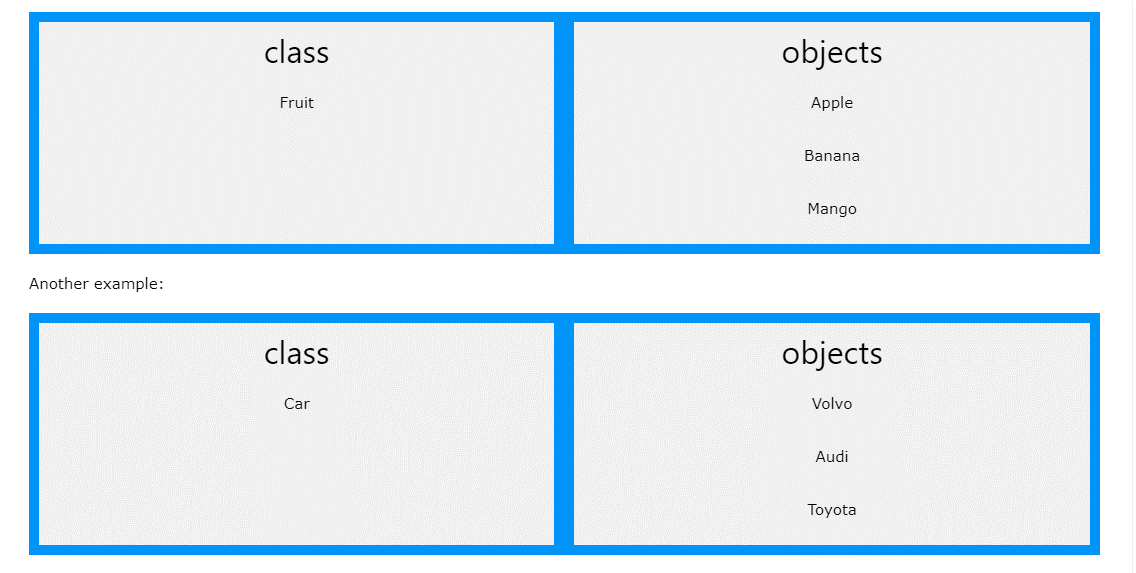
1. **What is OOP Concept?**

* OOPs refers to languages that use objects in programming, they use objects as a primary source to implement what is to happen in the code.
* It is a programming model or approach is ideal for the programs large and complex codes and needs to be actively updated and maintained. The programs are organized around objects rather than logic and functions.
* Object-oriented programming is used for designing a program using classes and objects.

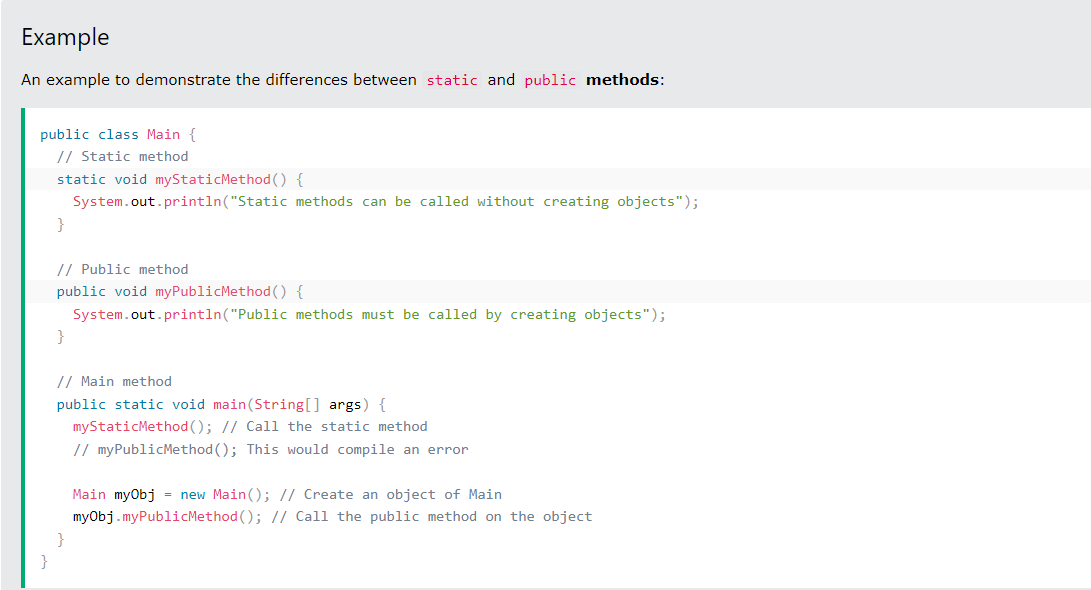
1. **What is class and object?**

* Class is a template where you can create the instance. A class is our working area which includes all our data. It contains variables and methods.
* Object is an instance of the class having instance variables. An object is created using by the new keyword in java.



1. **Static vs Non Static Methods**

* static method can be accessed without creating an object of the class.
* public method can only be accessed by objects.

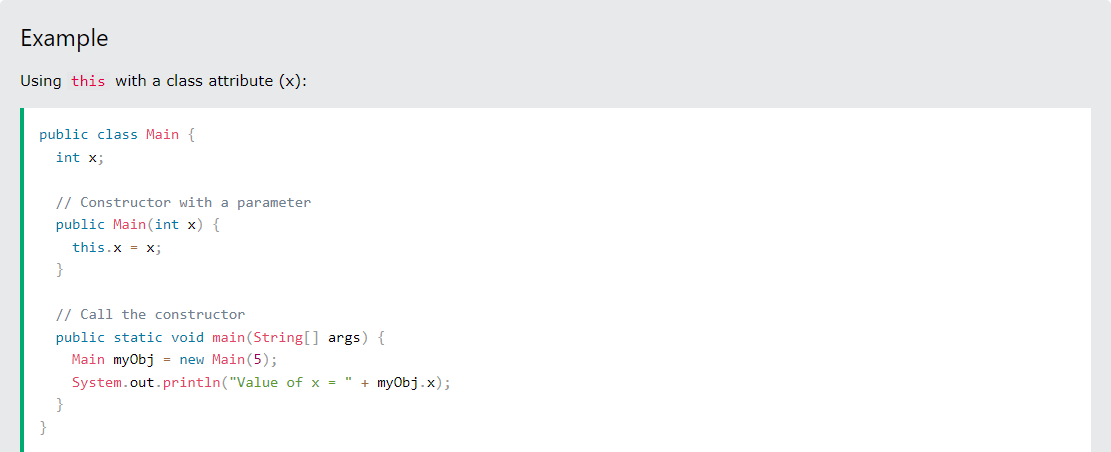


**What is Constructors?**

* A constructor in Java is a special method that is used to initialize objects.
* Constructor name must match the class name, and it cannot have a return type (like void).
* If we would like to create an object, we must use constructor.
* If constructor does not work, we cannot create an object.
* Constructor produce what we want. If we do not give a discription to construction, it brings us what it has.
* When we create a constructor, Java delete the default constructor.
* If we do not want to encounter a problem with our project, we must create a constractor (which does not have parameters) instead of default constructor.

-*this keyword*: The this keyword refers to the current object in a method or constructor.

* The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter).



If you do not add the keyword in the example above, the output would be "0" instead of "5".

1. **What is Java Modifiers?**

**Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

**Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

**Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

**Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

1. **What are the main concepts of OOPs in JAVA?**

* Inheritance, Encapsulation, Abstraction, Polymorphism.

**Inheritance:** parent child relationship between two classes. Child class can reuse all the codes written in parent class. It provides the code reusability.

-*Subclass (child)* - the class that inherits from another class. Subclass cannot reject properities of the superclass but can change the properities.

-*Superclass (parent)* - the class being inherited from. If we would like to use a class as parent, we need to make the modifier protected. If parent class member has a private access modifier, child class cannot use/able to access this member.

-To inherit from a class, use the extends keyword.

-Static methods and variables cannot be inherited.

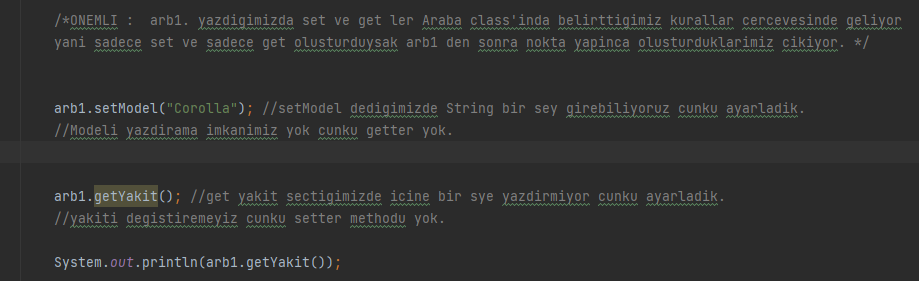
-If there is an extends keyword, there is a super contructer call. Super constructer is like default constructer. Even if we do not see it, it is exist. If we write a new constructer call on the first line, Java deletes the super constructer call.

-If there is no extends keyword, Java cannot accept super constructer call.

-If you don't want other classes to inherit from a class, use the final keyword:

**Encapsulation: Hiding data ...**by using **private**keyword, access private data by using getter and setter method.

* As in encapsulation, the data in a class is hidden from other classes using the data hiding concept which is achieved by making the members or methods of a class private
* Encapsulation is achieved by declaring the variables as private and providing public setter and getter methods to modify and view the variable values.
* The get method returns the variable value, and the set method sets the value via objects.



**Abstraction:** it is a methodology of hidding implementation details from the user. Abstraction lets you focus on what the object does instead of how it does it.

- Data abstraction is the process of hiding certain details and showing only essential information to the user. Abstraction can be achieved with either abstract classes or interfaces. The abstract keyword is a non-access modifier, used for classes and methods:

*- Abstract class:* is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

*- Abstract method:* can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

- An abstract class can have both abstract and regular methods.

-Abstract classes specify mandatory properties that child classes should have (like the standards in our daily life). All child classes that have abstract parent must override all abstract methods in the parent abstract class.

**Polymorphism:** it is the ability of a variable to take multiple forms. We use method overloading and overriding to achieve Polymorphism.

* For example, think of a superclass called Animal that has a method called animalSound(). Subclasses of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.):

1. **What is Type Casting?**

Type casting is when you assign a value of one primitive data type to another type.In Java, there are two types of casting:

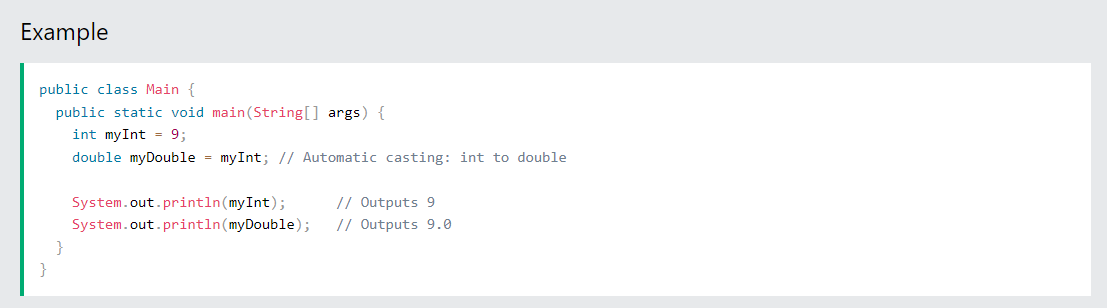
*Widening Casting (automatically)* - converting a smaller type to a larger type size

byte -> short -> char -> int -> long -> float -> double

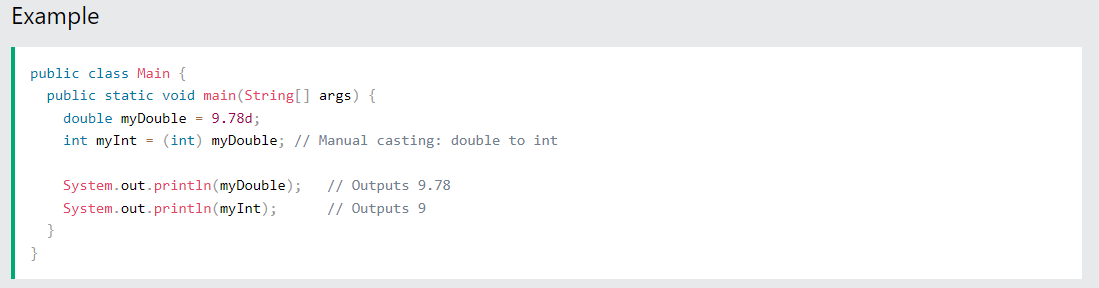
*Narrowing Casting (manually)* - converting a larger type to a smaller size type

double -> float -> long -> int -> char -> short -> byte

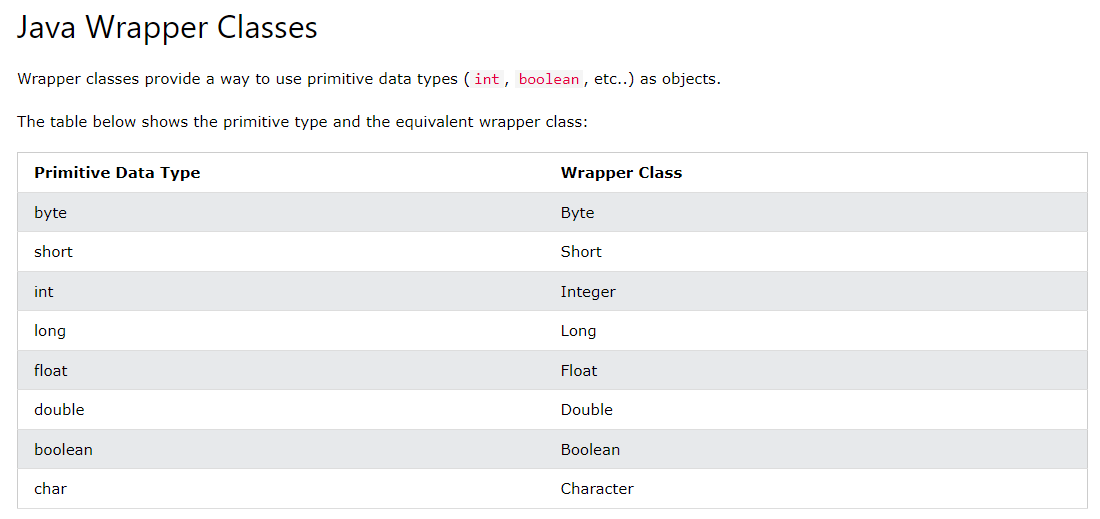
*Widening Casting*



Narrowing casting

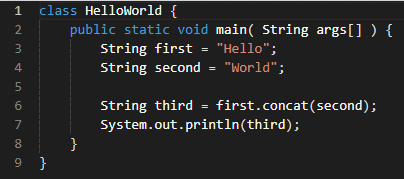
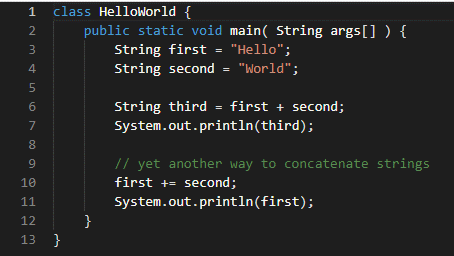


1. **What is Wrapper Classes?**



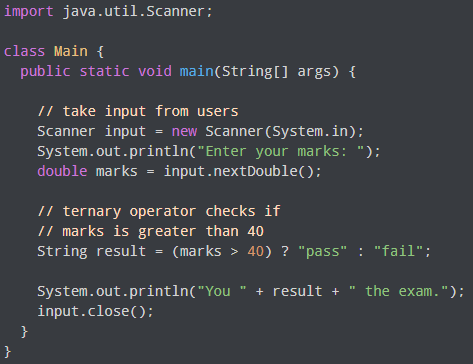
1. **What is concatenation?**

The concat() method appends (concatenate) a string to the end of another string.



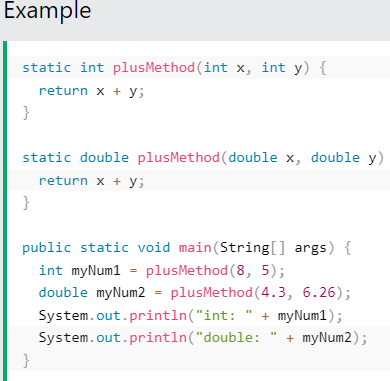
1. **What is Ternary Operator in Java?**

Here, condition is evaluated and if condition is true, expression1 is executed. And, if condition is false, expression2 is executed. The ternary operator takes 3 operands (condition, expression1, and expression2). Hence, the name ternary operator.



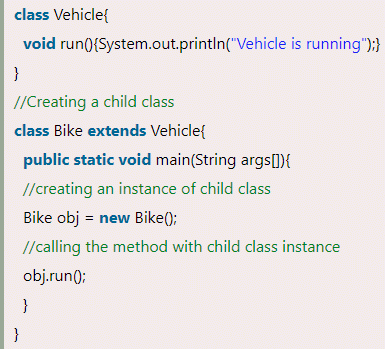
1. **What is Method Overloading?**

With method overloading, multiple methods can have the same name with different parameters. Instead of defining two methods that should do the same thing, it is better to overload one. In the example below, we overload the plusMethod method to work for both int and double:



1. **What is Overriding**

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java. In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.



1. **Diffference of overloading and overriding?**

When two or more methods in the same class have the same name but different parameters, it’s called Overloading. When the method signature (name and parameters) are the same in the superclass and the child class, it’s called Overriding.



**NOTE:** Private methods cannot be overridden. If you create a method in the Child class with the same signature as the private method in the parent class, this is NOT an overriding method.

**@Override notasyonu**

The @Override notation gives Java a task. With this notation, Java constantly checks two methods that are linked to each other. If you delete the overridden method in the parent class, it gives a CTE. It is not mandatory to use the @Override notation. If the overridding method is deleted and if we want that the code should gives us a CTE we should use the @Override notation.

1. **What are types of variables?**

|  |  |  |
| --- | --- | --- |
| **Local Variable** | **Instance Variable** | **Static Variable** |
| Defined within a method or a code block | Defined outside a method at the class level | Defined outside a method at the class level |
| Is only accessible in the method/code block where it is declared | Is accessible throughout the class | Is accessible throughout the class |
| Requires to be initialized before it is used | Is given default value based on its data type, so does not require to be initialized before it is used | Is given default value based on its data type, so does not require to be initialized before it is used. |

**NOTE 1:** Assigning a value to class variables is up to us. If we would like to assing a value, we can assign. If we do not assing a value to class variables, Java assings a value to these variables automatically.

For int, the deault value is 0

For String, the default value is null

For boolean, the default value is false

For char, the default value is “”(nothingness).

**NOTE 2:** Since instance variable is not a static variable, we **cannot** use it directly in the **static main methods**.

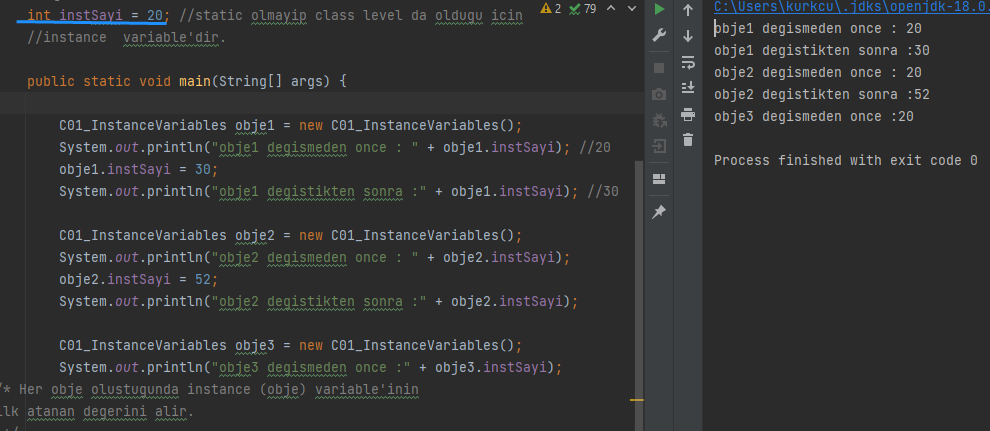
Instance variable is also **named as object** variable, therefore **we can use it when we create an object**.

We **can use** instance variables directly **in the non-static methods in the class**.

**NOTE 3:** Static variables can be accessed from anywhere within the class. All methods, whether static or not, can see and change static variables.

**NOTE 4:** To find the value of an instant variable, we must follow the code from the creation of the object to the desired line. In static variable, start at the beginning of the class and follow the code until the desired line. We need to find the last value of the static variable.

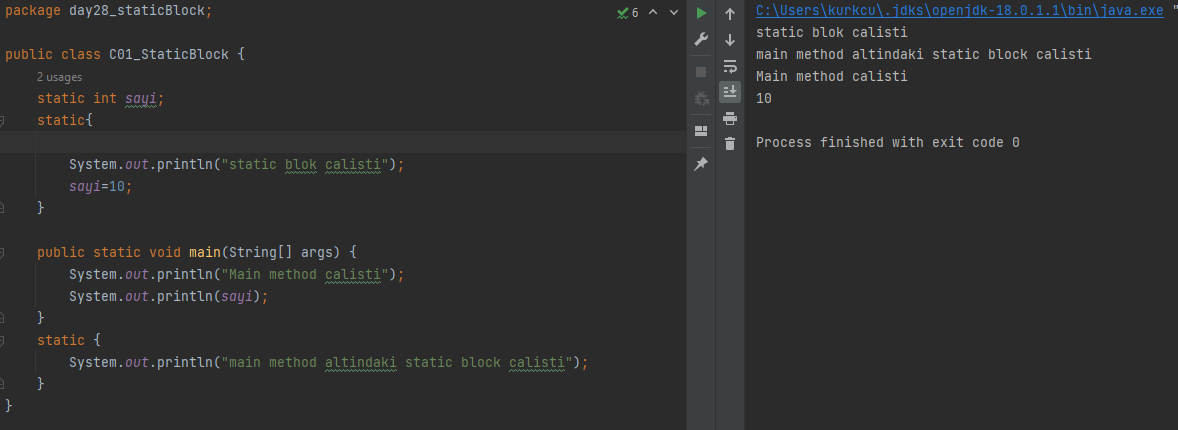
**NOTE 5:** When each object is created, the value of the previously created instance variable remains with the first assigned value.



**NOTE 6:** Static block runs before all its members (even before main method).

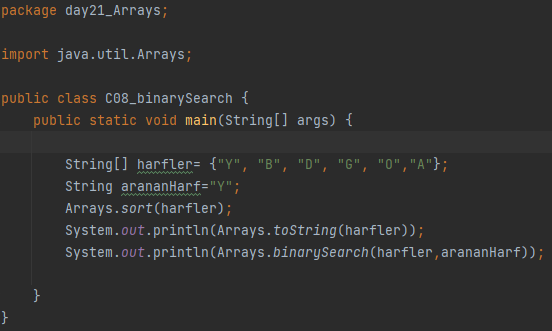
Static block works when the class is created, and is usually used to pre-configure the class or assign values to static variables.

It doesn't matter where the static blocks are in the class, static blocks work first. If there is more than one static block, it works from top to bottom.

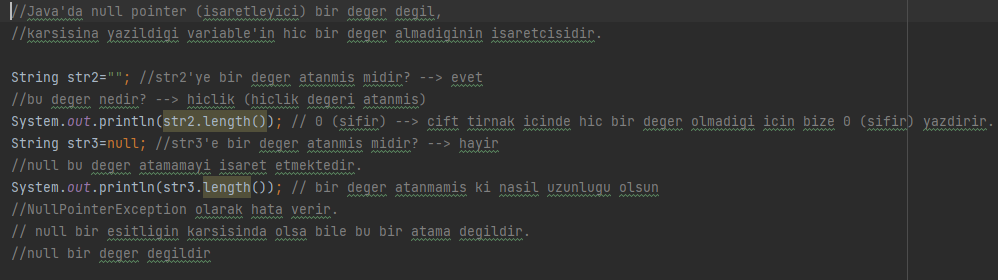


1. **What is Binary Search?**

Binary search is a shortcut for searching elements in java. Binary Search returns the index of the element we are looking for. However, for binary search to work, the array must first be sorted. If you do binarySearch without sorting, we may not find the result or we may get it wrong.



**13. What is Null Pointer?**



**14. Warrargs in Arrays**

**int...sayi**  bu gosterim integer variable'lardan olusan bir varargs demektir. Varargs array alt yapisini kullanir. Varargs: element sayisi uzun olan array.

