Class and Object

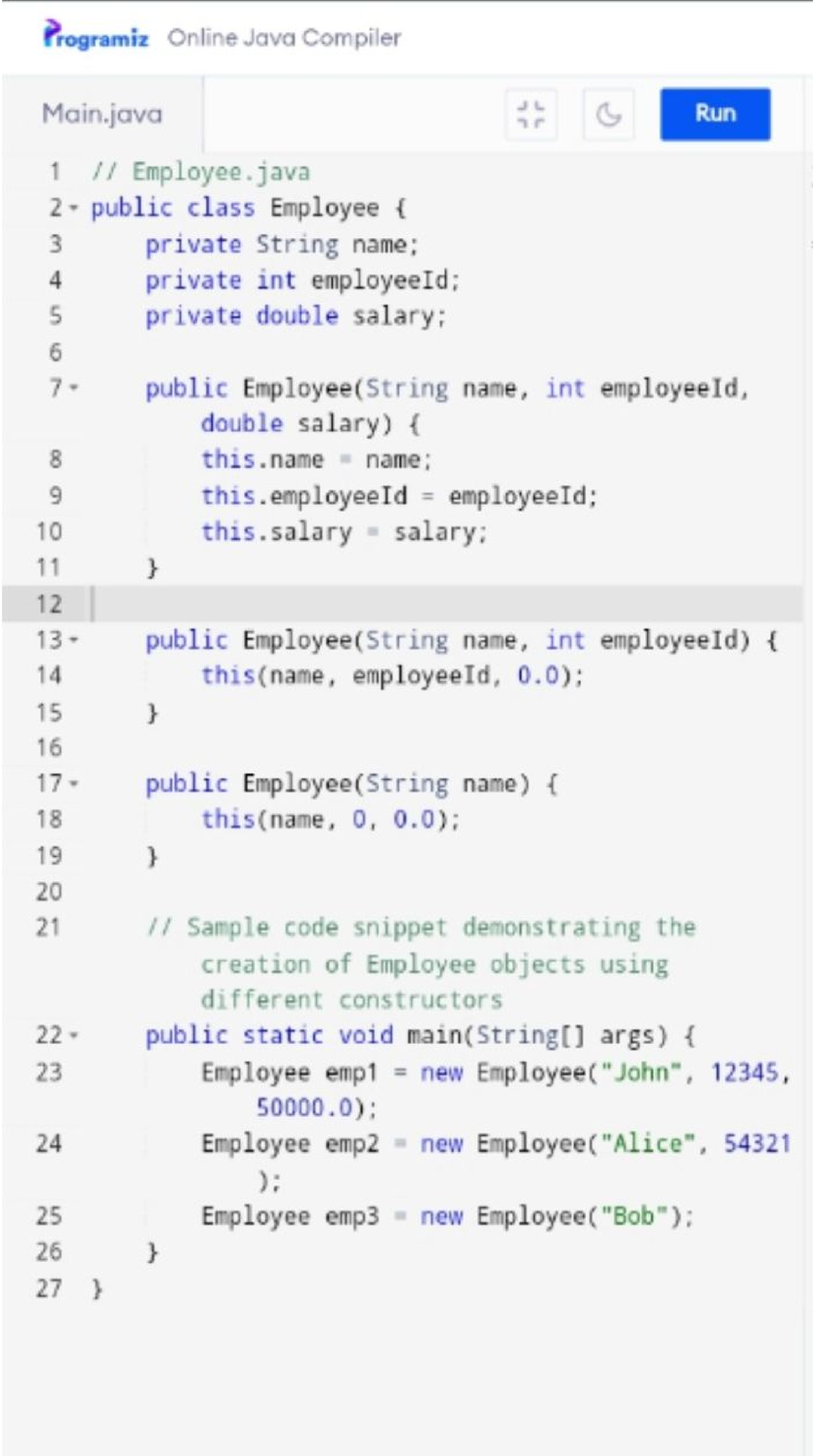
Create a Java class named `Employee` with private attributes for name, employee ID, and salary. Implement multiple constructors to initialize these attributes using constructor overloading. Provide a sample code snippet demonstrating the creation of `Employee` objects using different constructors.

Design a Java class called `Rectangle` with private attributes for length and width. Implement constructor overloading to initialize the rectangle's dimensions either by providing both length and width or by a single parameter to create a square. Provide an example of how to use these constructors to create `Rectangle` objects.

Develop a Java class named `BankAccount` with private attributes for account number, account holder name, and balance. Implement constructors to allow the creation of a bank account with just an account number, a combination of account number and holder name, and a full initialization with all attributes. Showcase the usage of these constructors in a concise code example.

Create a Java class called `Car` with private attributes for make, model, and year. Implement a default constructor and an overloaded constructor that takes all three attributes as parameters. Demonstrate how to use these constructors to instantiate `Car` objects with different initialization scenarios.

Define a Java class named `Student` with private attributes for student ID, name, and age. Implement multiple constructors to allow the creation of a student with only an ID, with an ID and name, and with all three attributes. Provide a clear example illustrating the usage of these constructors to create instances of the `Student` class.



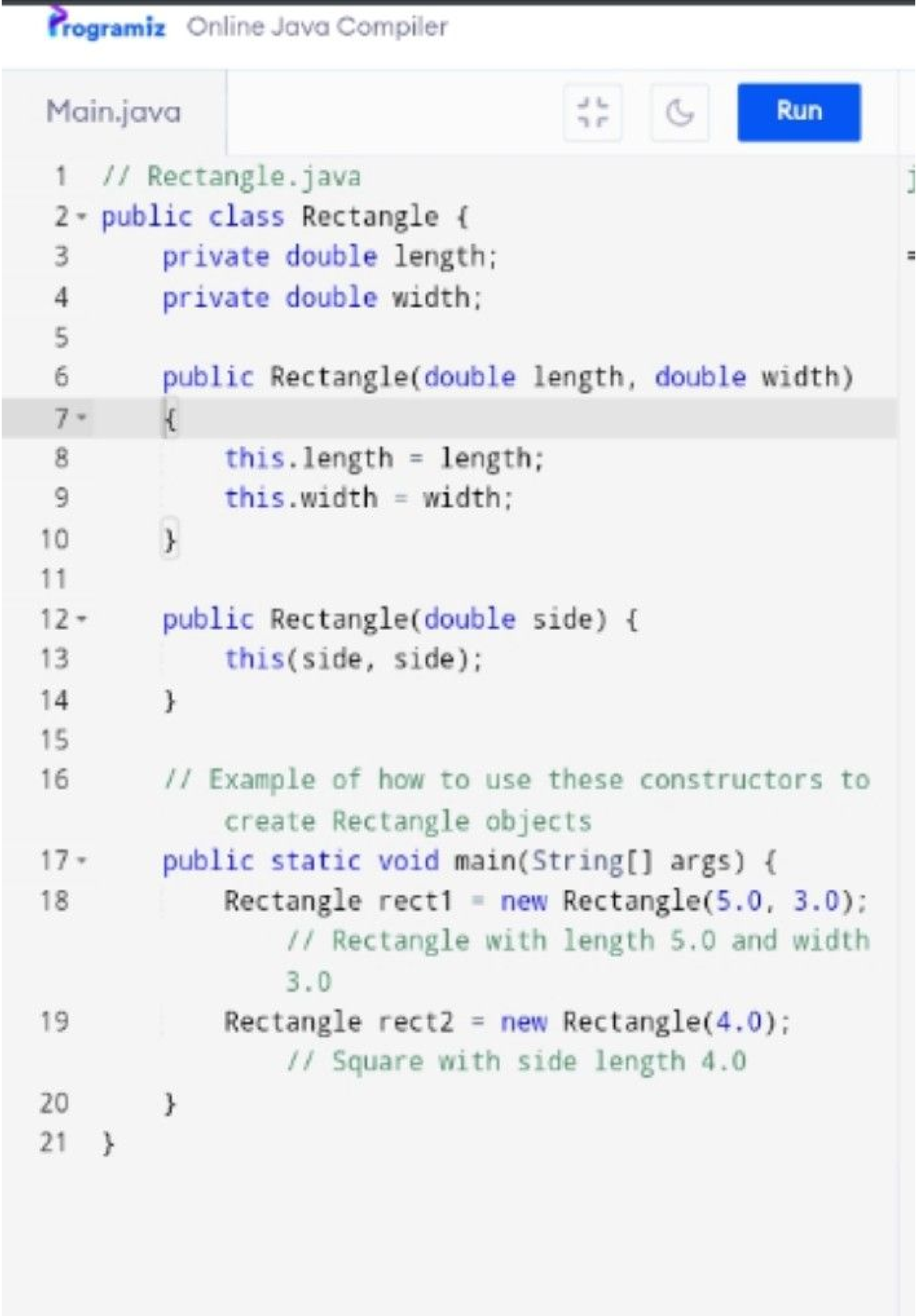
```java
// Employee.java
public class Employee {
    private String name;
    private int employeeId;
    private double salary;

    public Employee(String name, int employeeId, double salary) {
        this.name = name;
        this.employeeId = employeeId;
        this.salary = salary;
    }

    public Employee(String name, int employeeId) {
        this(name, employeeId, 0.0);
    }

    public Employee(String name) {
        this(name, 0, 0.0);
    }

    // Sample code snippet demonstrating the creation of Employee objects using different constructors
    public static void main(String[] args) {
        Employee emp1 = new Employee("John", 12345, 50000.0);
        Employee emp2 = new Employee("Alice", 54321);
        Employee emp3 = new Employee("Bob");
    }
}
```

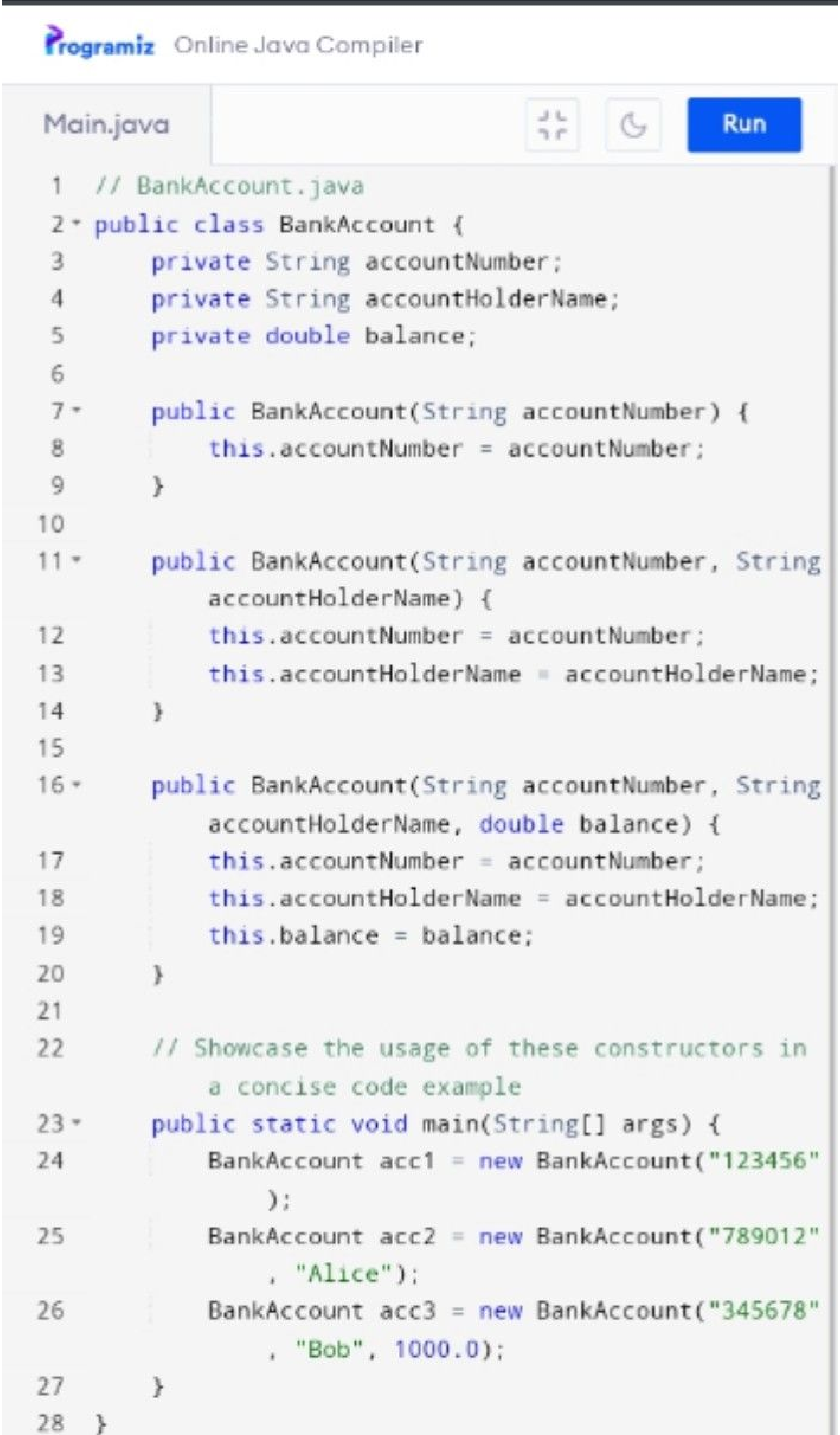


```java
// Rectangle.java
public class Rectangle {
    private double length;
    private double width;

    public Rectangle(double length, double width)
    {
        this.length = length;
        this.width = width;
    }

    public Rectangle(double side) {
        this(side, side);
    }

    // Example of how to use these constructors to create Rectangle objects
    public static void main(String[] args) {
        Rectangle rect1 = new Rectangle(5.0, 3.0); // Rectangle with length 5.0 and width 3.0
        Rectangle rect2 = new Rectangle(4.0); // Square with side length 4.0
    }
}
```



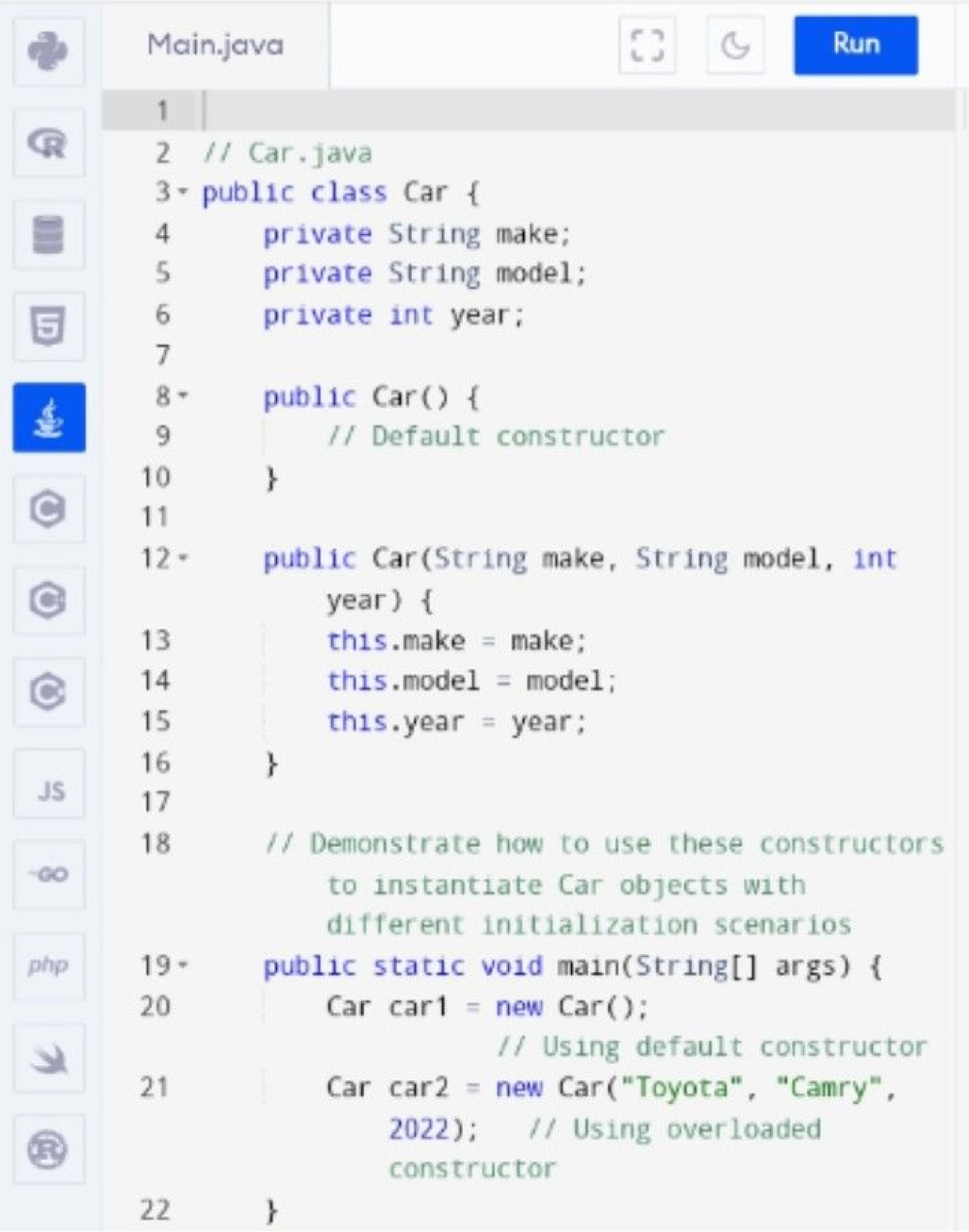
```java
// BankAccount.java
public class BankAccount {
    private String accountNumber;
    private String accountHolderName;
    private double balance;

    public BankAccount(String accountNumber) {
        this.accountNumber = accountNumber;
    }

    public BankAccount(String accountNumber, String accountHolderName) {
        this.accountNumber = accountNumber;
        this.accountHolderName = accountHolderName;
    }

    public BankAccount(String accountNumber, String accountHolderName, double balance) {
        this.accountNumber = accountNumber;
        this.accountHolderName = accountHolderName;
        this.balance = balance;
    }

    // Showcase the usage of these constructors in a concise code example
    public static void main(String[] args) {
        BankAccount acc1 = new BankAccount("123456");
        BankAccount acc2 = new BankAccount("789012", "Alice");
        BankAccount acc3 = new BankAccount("345678", "Bob", 1000.0);
    }
}
```



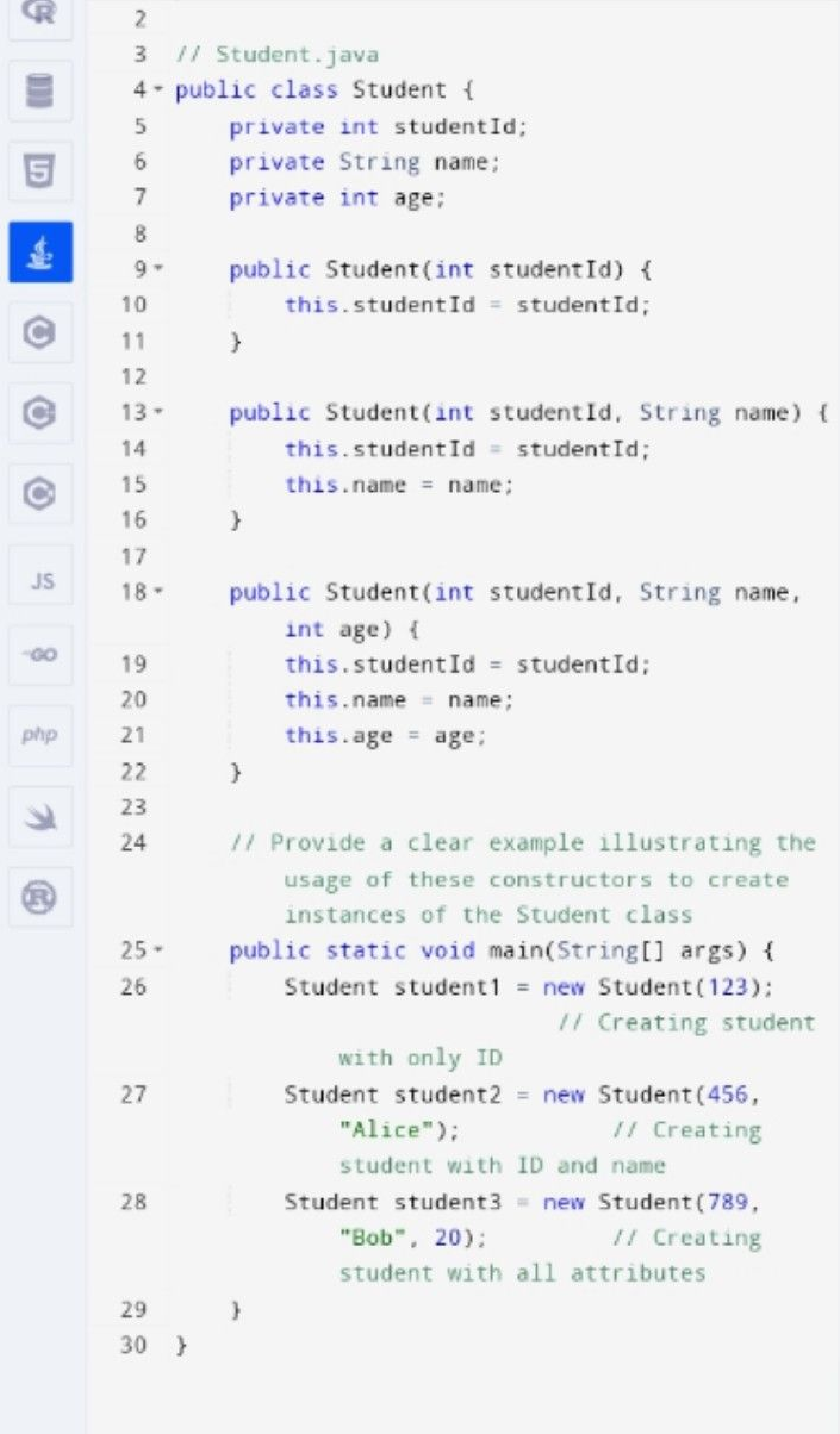
```java

// Car.java
public class Car {
    private String make;
    private String model;
    private int year;

    public Car() {
        // Default constructor
    }

    public Car(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }

    // Demonstrate how to use these constructors to instantiate Car objects with different initialization scenarios
    public static void main(String[] args) {
        Car car1 = new Car();          // Using default constructor
        Car car2 = new Car("Toyota", "Camry", 2022);   // Using overloaded constructor
    }
```



```java

// Student.java
public class Student {
    private int studentId;
    private String name;
    private int age;

    public Student(int studentId) {
        this.studentId = studentId;
    }

    public Student(int studentId, String name) {
        this.studentId = studentId;
        this.name = name;
    }

    public Student(int studentId, String name, int age) {
        this.studentId = studentId;
        this.name = name;
        this.age = age;
    }

    // Provide a clear example illustrating the usage of these constructors to create instances of the Student class
    public static void main(String[] args) {
        Student student1 = new Student(123);                    // Creating student with only ID
        Student student2 = new Student(456, "Alice");           // Creating student with ID and name
        Student student3 = new Student(789, "Bob", 20);         // Creating student with all attributes
    }
}
```