JAVA BROCODE

1.

Main.java

```java
package brocodejava;

public class Main {

    public static void main(String[] args) {

        System.out.println("I am Vishal \n");
        System.out.println("I am from SSN");
        System.out.println(); //sysout ctrl+space
    }

}
```

2. Variables

A placeholder for a value that behaves as the value it contains

data type	size	primitive/ reference	value
★ boolean	1 bit	primitive	true or false
byte	1 byte	primitive	-128 to 127
short	2 bytes	primitive	-32,768 to 32,767
★ int	4 bytes	primitive	-2 billion to 2 billion
long	8 bytes	primitive	-9 quintillion to 9 quintillion
float	4 bytes	primitive	fractional number up to 6-7 digits ex. 3.141592f
★ double	8 bytes	primitive	fractional number up to 15 digits ex. 3.141592653589793
★ char	2 bytes	primitive	single character/letter/ASCII value ex. 'f'
★ String	varies	reference	a sequence of characters ex. "Hello world!"

primitive vs reference

- 8 types (boolean,byte,etc.)
- stores data
- can only hold 1 value
- less memory
- fast

- unlimited (user defined)
- stores an address
- could hold more than 1 value
- more memory
- slower

variable → declaration + assignment or initialisation

Main.java

```java
package brocodejava;

public class Main {

    public static void main(String[] args) {

        int x; //declaration
        x = 123; //assignment
        int y = 1234; //initialisation
        System.out.println(x);
        System.out.println(y);
    }

}
```

Main.java

```java
package brocodejava;

public class Main {

    public static void main(String[] args) {

        long x = 1234567890L;
        int y = 123;
        float z = 3.14f;
        boolean f = false;

        char symbol = '@';
        String name = "Vishal is atomic";

        System.out.println("The number is "+x);
        System.out.println("The number is "+y);
        System.out.println("The number is "+z);
        System.out.println(f);
        System.out.println(symbol);
        System.out.println("I know that "+name);
    }

}
```

3. Swap two variables

*Three.java

```java
package brocodejava;

public class Three {

    public static void main(String[] args) {
        String x = "water";
        String y = "Acid";
        String temp = null;

        temp = x;
        x=y;
        y=temp;

        System.out.println("x: "+x);
        System.out.println("y: "+y);
    }

}
```

4. User Input

Four.java

```java

package brocodejava;
import java.util.Scanner;
public class Four {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.println("What is your name? ");

        String name = scanner.nextLine();
        System.out.println("How old are you? ");
        int age = scanner.nextInt();
        scanner.nextLine(); //to clear the contents in the scanner
        System.out.println("What is your favourite food?");
        String food = scanner.nextLine();

        System.out.println("Hello "+name);
        System.out.println("You are "+age+" years old");
        System.out.println("You like "+food);
        scanner.close();

    }

}

```

```
<terminated> Four [Java Application]
What is your name?
Vishal
How old are you?
21
What is your favourite food?
lassi
Hello Vishal
You are 21 years old
You like lassi
```

5. Expressions

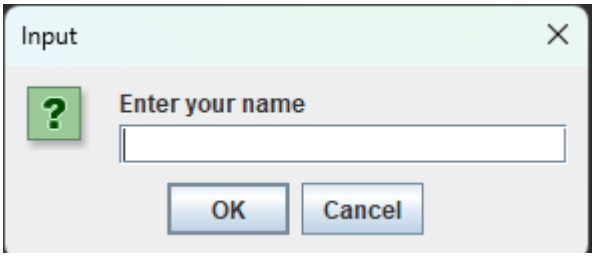
Five.java

```java
package brocodejava;

public class Five {

    public static void main(String[] args) {
        // expression = operands & operators
        // operands = values, variables, numbers, quantity
        // operators = +-*/%

//      int num = 10;
//
//      num = num/3;

        double num = 10;
        num = (double) num/3;

        System.out.println(num);

    }

}
```

6. GUI intro

Six.java

```java
package brocodejava;
import javax.swing.JOptionPane;
public class Six {

    public static void main(String[] args) {
        String name = JOptionPane.showInputDialog("Enter your name ");

        JOptionPane.showMessageDialog(null, "Hello "+name);

        int age = Integer.parseInt(JOptionPane.showInputDialog("Enter your age: "));
        JOptionPane.showMessageDialog(null, "Your age is "+age);

        double height = Double.parseDouble(JOptionPane.showInputDialog("Enter your height: "));
        JOptionPane.showMessageDialog(null, "Your height is "+height);

    }

}
```



7. Math Class

Seven.java

```java
package brocodejava;
import java.util.Scanner;
public class Seven {

    public static void main(String[] args) {

//      double x = 3.14;
//      double y = -10;
//      double z = Math.max(x,y);
//      double z = Math.min(x,y);
//      double z = Math.abs(y);
//      double z = Math.sqrt(x);
//      double z = Math.round(x);
//      double z = Math.ceil(x);
//      double z = Math.floor(x);
//      System.out.println(z);

        double x;
        double y;
        double z;

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter side x: ");
        x = scanner.nextDouble();
        System.out.println("Enter side y: ");
        y = scanner.nextDouble();

        z = Math.sqrt((x*x)+(y*y));
        System.out.println("Hypotenuse: "+z);

    }

}
```

8. Random Numbers

Eight.java

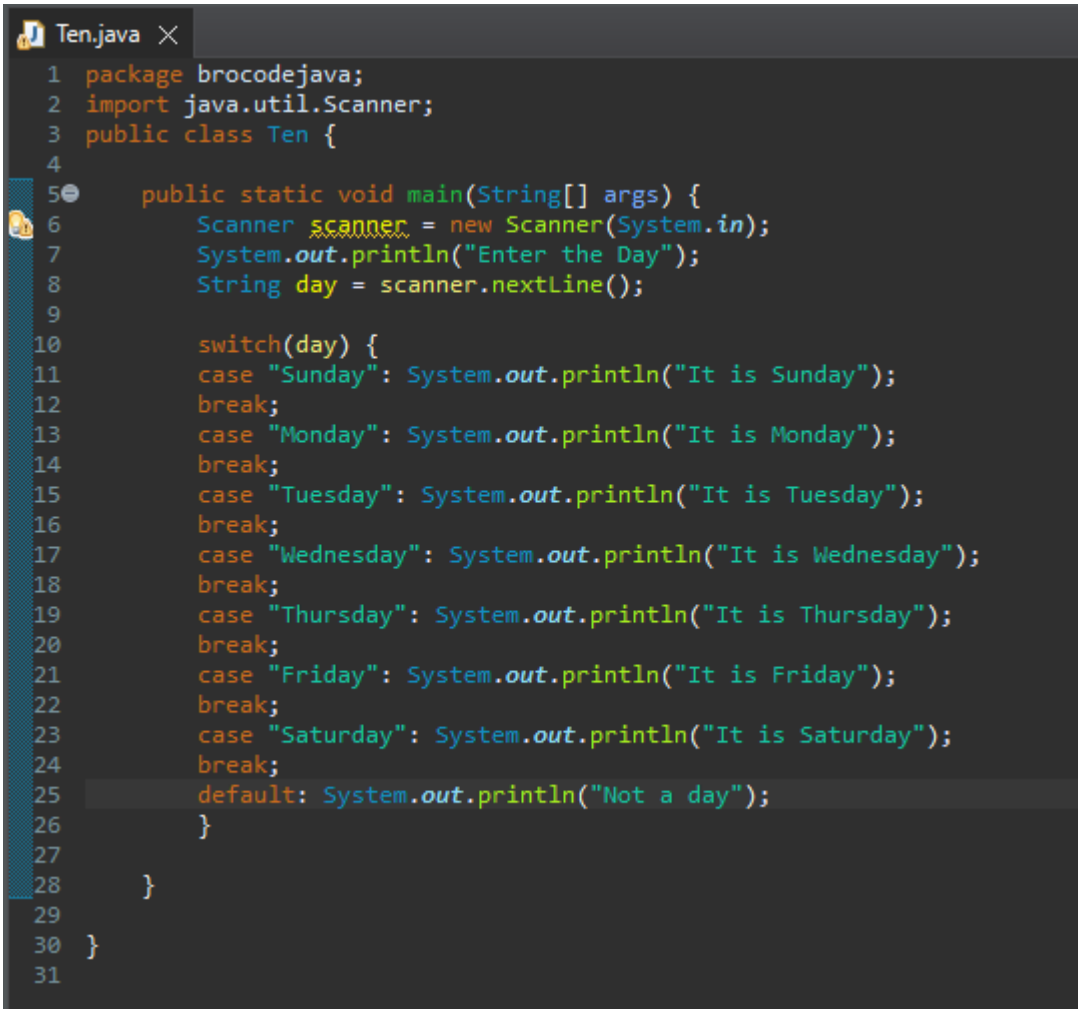
```java
package brocodejava;
import java.util.Random;
public class Eight {

    public static void main(String[] args) {

        Random random = new Random();

//      int x = random.nextInt(); (Result will be in the range -2billion to positive 2billion)
//      int x = random.nextInt(6)+1;

//      double y = random.nextDouble(); (Random value between 0 and 1)
        boolean z = random.nextBoolean();

        System.out.println(z);
    }

}
```

9. if statements

Nine.java

```java
package brocodejava;

public class Nine {

    public static void main(String[] args) {

        int age = 80;

        if (age>=75) {
            System.out.println("Ok Boomer");
        }
        else if(age>=18) {
            System.out.println("You are an adult");
        }
        else {
            System.out.println("You are not an adult");
        }

    }

}
```

10. switches



11. Logical Operators

Eleven.java

```java
package brocodejava;

public class Eleven {

    public static void main(String[] args) {
//      && = AND : Both conditions must be true
//      || = OR : either condition must be true
//      ! = NOT : reverses boolean value of a condition

        int temp = 25;

        if (temp>30) {
            System.out.println("It is hot outside");
        }
        else if(temp>=20 && temp<=30) {
            System.out.println("It is warm outside");
        }
        else {
            System.out.println("It is cold outside");
        }

    }

}

```

```java

        Scanner scanner = new Scanner(System.in);

        System.out.println("You are playing a game! Press q or Q to quit");
        String response = scanner.next();

        if(!response.equals("q") && !response.equals("Q")) {
            System.out.println("YOu are still playing the game *pew pew*");
        }
        else {
            System.out.println("You quit the game");
        }

    }
```

Problems Javadoc Declaration Console Coverage Call Hierarchy

<terminated> Main (6) [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Oct 14, 2020, 4:26:15 PM)

```
You are playing a game! Press q or Q to quit
Q
You quit the game
```

12. while loop

Twelve.java

```java
package brocodejava;
import java.util.Scanner;
public class Twelve {

    public static void main(String[] args) {
        // as long as the condition is true
        Scanner scanner = new Scanner(System.in);
        String name = "";

//      while(name.isBlank()) {
//          System.out.println("Enter your name: ");
//          name = scanner.nextLine();
//      }

        do {
            System.out.println("Enter your name:");
            name = scanner.nextLine();
        } while (name.isBlank());


        System.out.println("Hello "+name);

    }

}
```

13. For Loop

Thirteen.java

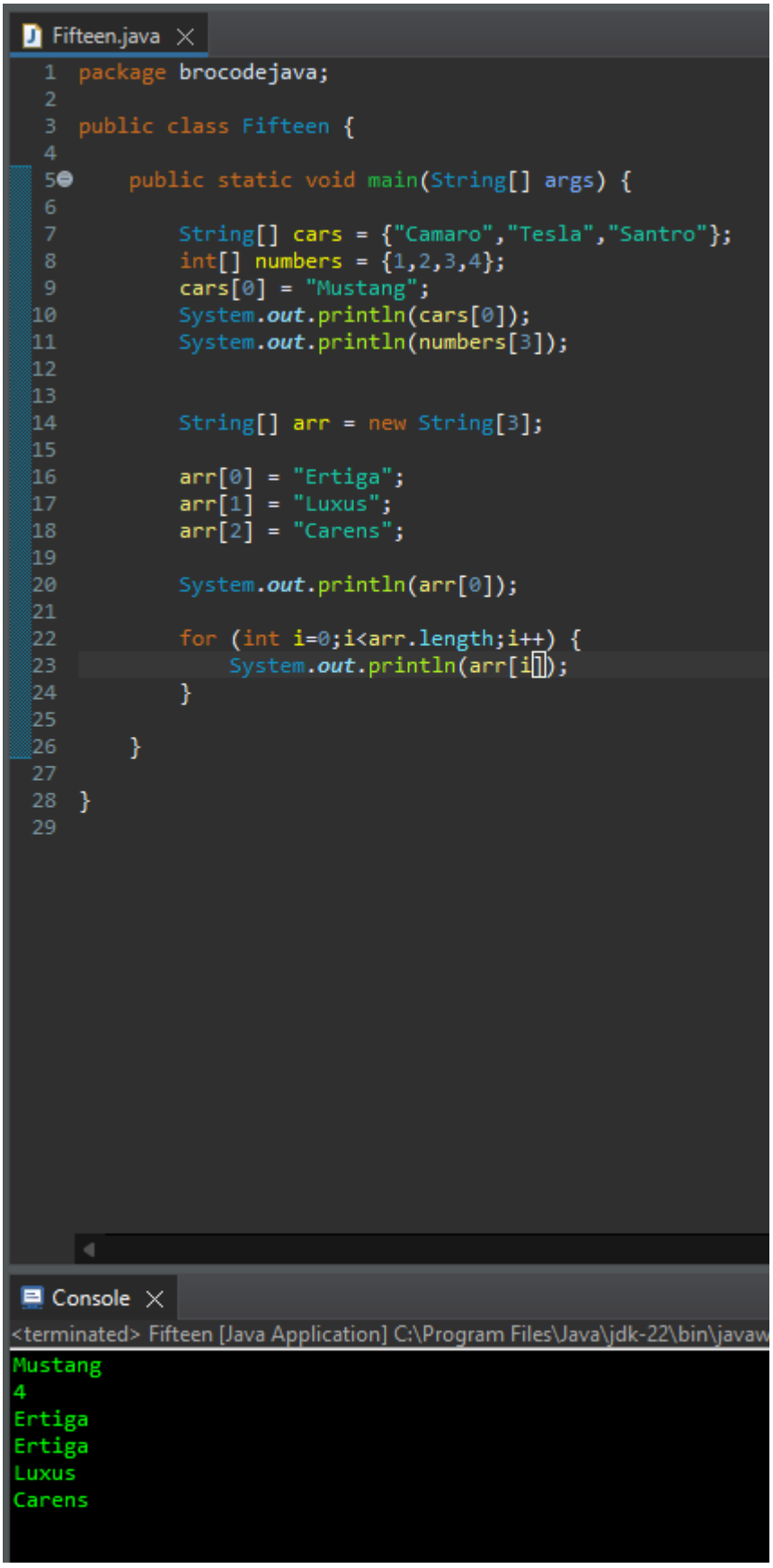
```java
package brocodejava;
import java.util.Scanner;
public class Thirteen {

    public static void main(String[] args) {

        for (int i = 0; i<=10;i++) {
            System.out.println(i);
        }

    }

}
```

14. Nested Loops

Fourteen.java

```java
package brocodejava;
import java.util.Scanner;
public class Fourteen {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        int rows;
        int columns;
        String symbol = "";

        System.out.println("Enter number of rows");
        rows = scanner.nextInt();

        System.out.println("Enter number of cols");
        columns = scanner.nextInt();

        System.out.println("Enter the symbol");
        symbol = scanner.next();

        for (int i=1;i<=rows;i++) {
            System.out.println();
            for (int j=1;j<=columns;j++) {
                System.out.print(symbol);
            }
        }

    }

}
```

15. Arrays


Fifteen.java
1 package brocodejava;
2
3 public class Fifteen {
4
5 public static void main(String[] args) {
6
7 String[] cars = {"Camaro","Tesla","Santro"};
8 int[] numbers = {1,2,3,4};
9 cars[0] = "Mustang";
10 System.out.println(cars[0]);
11 System.out.println(numbers[3]);
12
13
14 String[] arr = new String[3];
15
16 arr[0] = "Ertiga";
17 arr[1] = "Luxus";
18 arr[2] = "Carens";
19
20 System.out.println(arr[0]);
21
22 for (int i=0;i<arr.length;i++) {
23 System.out.println(arr[i]);
24 }
25
26 }
27
28 }
29
Console
<terminated> Fifteen [Java Application] C:\Program Files\Java\jdk-22\bin\javaw
Mustang
4
Ertiga
Ertiga
Luxus
Carens

16. 2D Arrays

Sixteen.java

```java
package brocodejava;

public class Sixteen {

    public static void main(String[] args) {

        // 2D arrays = array of arrays

        String[][] cars = new String[3][3];

        cars[0][0] = "Camaro";
        cars[0][1] = "Corvette";
        cars[0][2] = "Ertiga";
        cars[1][0] = "Alto";
        cars[1][1] = "Ritz";
        cars[1][2] = "Nano";
        cars[2][0] = "Carnival";
        cars[2][1] = "Carens";
        cars[2][2] = "Mustang";

        for (int i=0; i<cars.length;i++) {
            System.out.println();
            for (int j=0;j<cars[i].length;j++) {
                System.out.print(cars[i][j]+" ");
            }

        String[][] cars1 = {{"Camaro","Corvette","Ertiga"},
                            {"Alto","Ritz","Nano"},
                            {"Carnival","Carens","Mustang"}};

        System.out.println();
        for (int i1=0; i1<cars1.length;i1++) {
            System.out.println();
            for (int j=0;j<cars1[i1].length;j++) {
                System.out.print(cars1[i1][j]+" ");
            }
        }

    }

}
}
```

Console

```
<terminated> Sixteen [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (19 Au
Alto Ritz Nano
Carnival Carens Mustang
Alto Ritz Nano

Camaro Corvette Ertiga
Alto Ritz Nano
Carnival Carens Mustang
Carnival Carens Mustang

Camaro Corvette Ertiga
Alto Ritz Nano
Carnival Carens Mustang
```

17. String Methods

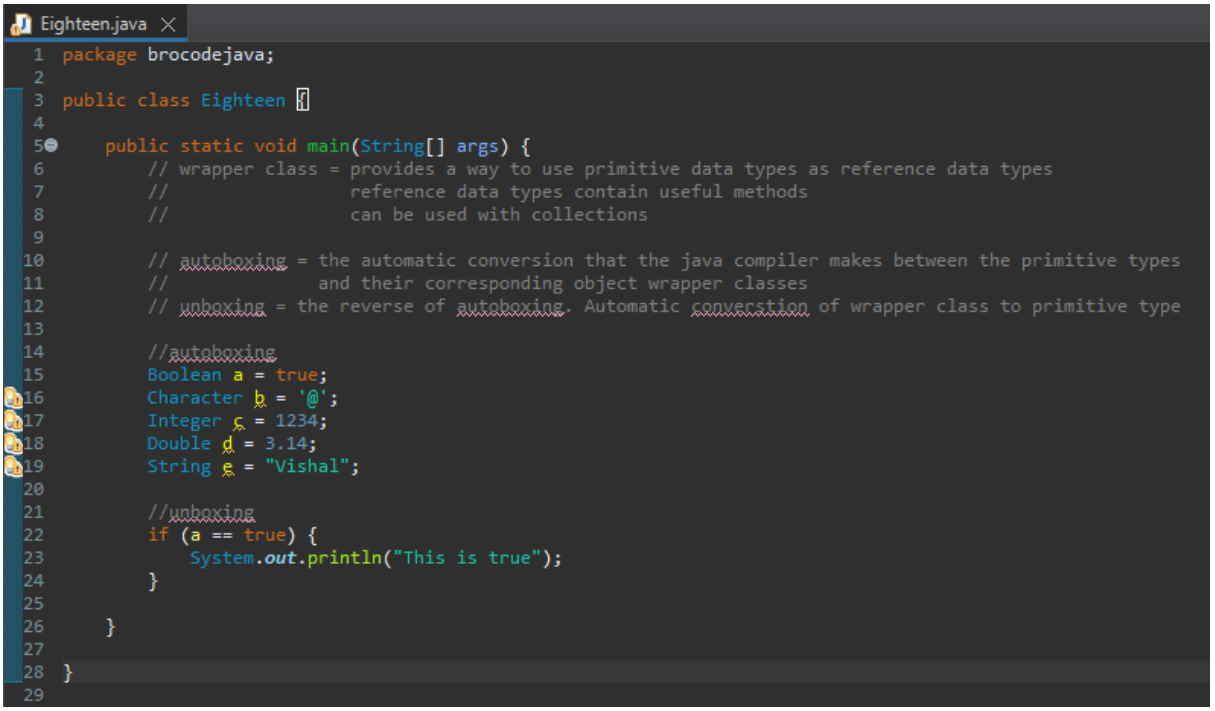
*Seventeen.java

```java
public class Seventeen {

    public static void main(String[] args) {
        // String = a reference data type that can store one or more characters
        //          reference data types have access to useful methods
        String name = "Vishal";
        boolean result = name.equals("Vishal");
        boolean result1 = name.equals("vishal");
        boolean result2 = name.equalsIgnoreCase("vishal");

        if (name.equals("  Vishal  ")) {
            System.out.println("True Vishal Real");
        }
        System.out.println(result);
        System.out.println(result1);
        System.out.println(result2);
        System.out.println();

        int result3 = name.length();
        System.out.println(result3);
        System.out.println();

        char result4 = name.charAt(3);
        System.out.println(result4);
        System.out.println();

        int result5 = name.indexOf('a');
        System.out.println(result5);
        System.out.println();

        int result6 = name.indexOf('z');
        System.out.println(result6);
        System.out.println();

        boolean result7 = name.isEmpty();
        System.out.println(result7);
        System.out.println();

        String result8 = name.toUpperCase();
        System.out.println(result8);

        String result9 = name.toLowerCase();
        System.out.println(result9);
        System.out.println();

        String result10 = name.trim();
        System.out.println(result10);
        System.out.println();

        String result11 = name.replace("Vi","Rizz");
        System.out.println(result11);

```

```
true
false
true

6

h

4

-1

false

VISHAL
vishal

Vishal

Rizzshal
```

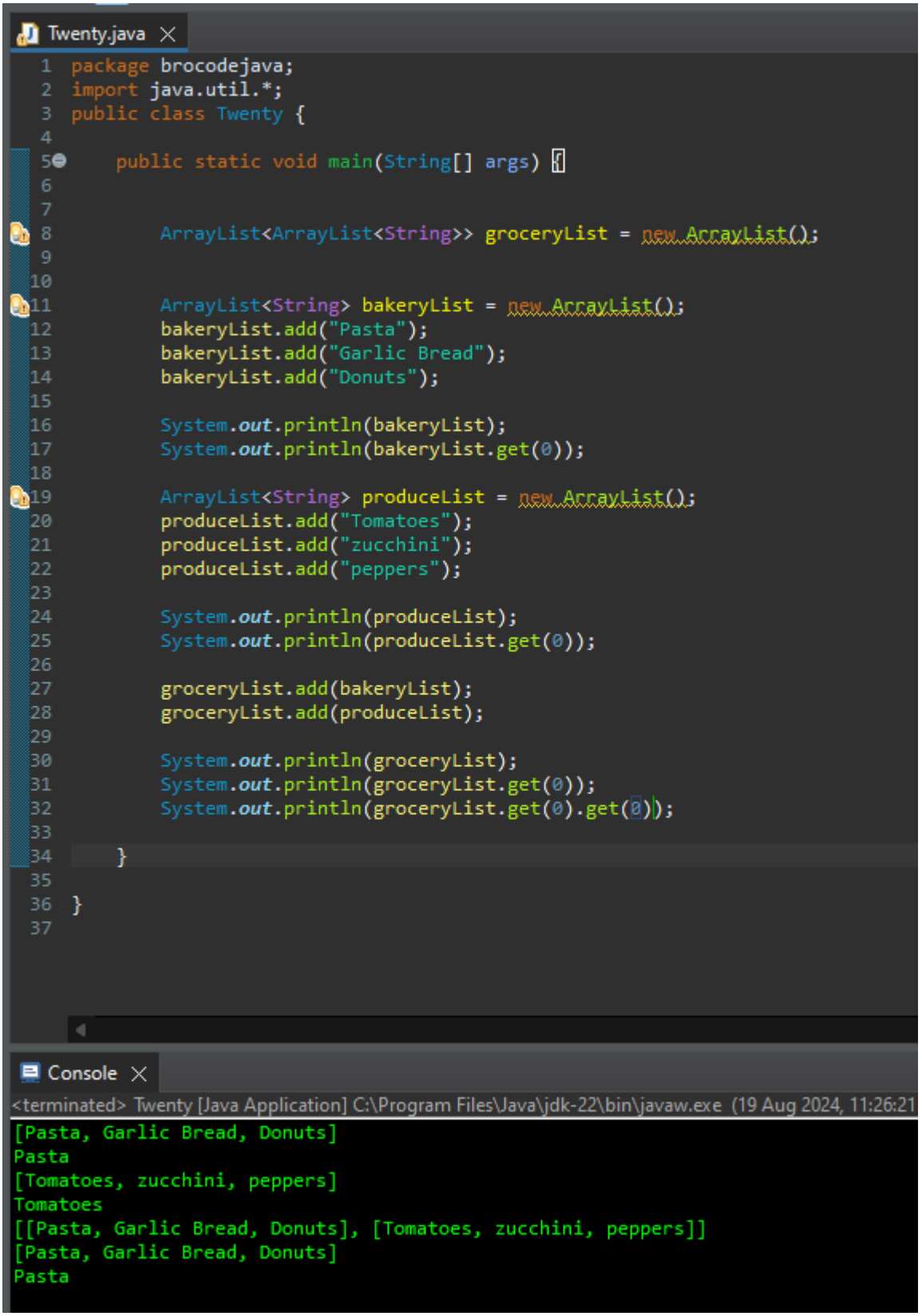
18. Wrapper Classes

```
// wrapper class =  provides a way to use primitive data types as reference data types
//                  reference data types contain useful methods
//                  can be used with collections (ex.ArrayList)

//primitive     //wrapper
//---------     //-------
// boolean      Boolean
// char         Character
// int          Integer
// double       Double

// autoboxing = the automatic conversion that the Java compiler makes between the primitive t
// unboxing = the reverse of autoboxing. Automatic conversion of wrapper class to primitive
```



19. ArrayList

20. 2D ArrayList



21. for-each loop

TwentyOne.java

```java
package brocodejava;

import java.util.ArrayList;

public class TwentyOne {

    public static void main(String[] args) {

        //String[] animals = {"cat","dog","rat","bird"};
        ArrayList<String> animals = new ArrayList<String>();
        animals.add("dog");
        animals.add("cat");
        animals.add("rat");
        animals.add("bird");
        for(String i : animals) {
            System.out.println(i);
        }

    }

}
```

22. Methods

TwentyTwo.java

```java
package brocodejava;

public class TwentyTwo {

    public static void main(String[] args) {
        // method = block of code which is executed whenever it is called upon
        String name = "Vishal";
        int age = 21;

        hello(name,age);


    }

    static void hello(String name, int age) {
        System.out.println("Hello "+name);
        System.out.println("Your are "+age);

    }

}
```

TwentyTwoagain.java

```java
package brocodejava;

public class TwentyTwoagain {

    public static void main(String[] args) {

        int x = 3;
        int y = 4;

        int z = add(x,y);
        System.out.println(z);
    }


    static int add(int x, int y) {
        int z = x+y;
        return z;
    }
}

```

23. Overloaded Methods

*TwentyThree.java

```java
package brocodejava;

public class TwentyThree {

    public static void main(String[] args) {
        // overload methods = methods that share the same name but have different parameters
        //                    method name + parameters = method signature
        //                    parameters =  number of parameters, data type and the order of parameters can be changed
        //                    and overload methods can be created with same name but different parameters

        int x = add(1,2);
        System.out.println(x);

        int y = add(1,2,3);
        System.out.println(y);

        int z = add(1,2,3,4);
        System.out.println(z);

        double x1 = add(1.1,2.2);
        System.out.println(x1);

        double y1 = add(1.1,2.2,3.3);
        System.out.println(y1);

        double z1 = add(1.1,2.2,3.3,4.4);
        System.out.println(z1);
    }
    static int add(int a,int b) {
        System.out.println("This is overload method #1");
        return a+b;
    }
    static int add(int a,int b,int c) {
        System.out.println("This is overload method #2");
        return a+b+c;
    }
    static int add(int a,int b,int c, int d) {
        System.out.println("This is overload method #3");
        return a+b+c+d;
    }
    static double add(double a,double b) {
        System.out.println("This is overload method #4");
        return a+b;
    }
    static double add(double a,double b,double c) {
        System.out.println("This is overload method #5");
        return a+b+c;
    }
    static double add(double a,double b,double c, double d) {
        System.out.println("This is overload method #6");
        return a+b+c+d;
    }

}
```

```
<terminated> TwentyThree [Java App
This is overload method #1
3
This is overload method #2
6
This is overload method #3
10
This is overload method #4
3.3000000000000003
This is overload method #5
6.6
This is overload method #6
11.0
```

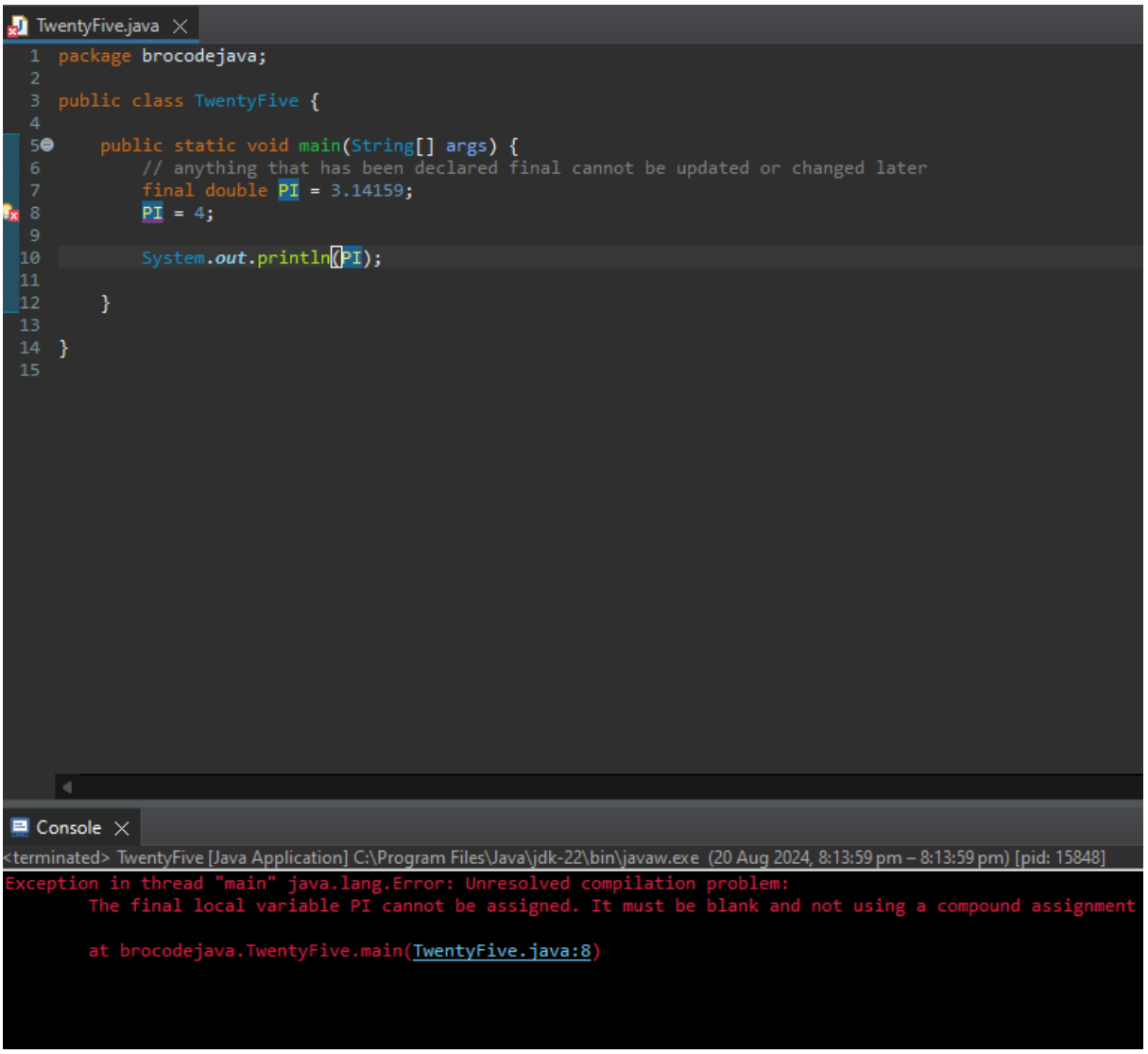
24. printf

TwentyFour.java

```java
public class TwentyFour {

    public static void main(String[] args) {
        // printf() = optional method to control,format and display test to the console window
        //          two arguments = format string + (object/variable/value)
        //          % [flags] [precision] [width] [conversion-character]

        System.out.printf("%s is a format string %d ","vishal",123);
        boolean myBoolean = true;
        char myChar = '@';
        String myString = "Vishal";
        int myInt = 50;
        double myDouble = 34589.145246;

        //[conversion-character]
        System.out.printf("%b",myBoolean);
        System.out.println();
        System.out.printf("%c",myChar);
        System.out.println();
        System.out.printf("%s",myString);
        System.out.println();
        System.out.printf("%d",myInt);
        System.out.println();
        System.out.printf("%f",myDouble);
        System.out.println();
        //[width]
        // minimum number of characters to be written as output
        System.out.printf("Hello %10s", myString);
        System.out.println();
        //[precision]
        // sets number of digits of precision when outputting floating-point values

        System.out.printf("You have %.2f money left",myDouble);
        System.out.println();
        System.out.printf("You have %f money left",myDouble);
        System.out.println();

        //[flags]
        // adds an effect to output based on the flag added to format specifier
        // - : left-justify
        // + : output a plus (+) or minus(-) sign for a numeric value
        // 0 : numeric values are zero-padded
        // , : comma grouping separator if numbers>1000
        System.out.println();
        System.out.printf("You have %-10f money left",myDouble);
        System.out.println();
        System.out.printf("You have %+f money left",myDouble);
        System.out.println();
        System.out.printf("You have %020f money left",myDouble);
        System.out.println();
        System.out.printf("You have %,f money left",myDouble);

    }
```

```
vishal is a format string 123 true
@
Vishal
50
34589.145246
Hello     Vishal
You have 34589.15 money left
You have 34589.145246 money left

You have 34589.145246 money left
You have +34589.145246 money left
You have 0000000034589.145246 money left
You have 34,589.145246 money left
```

25. final keyword



26. Object Oriented Programming (OOP

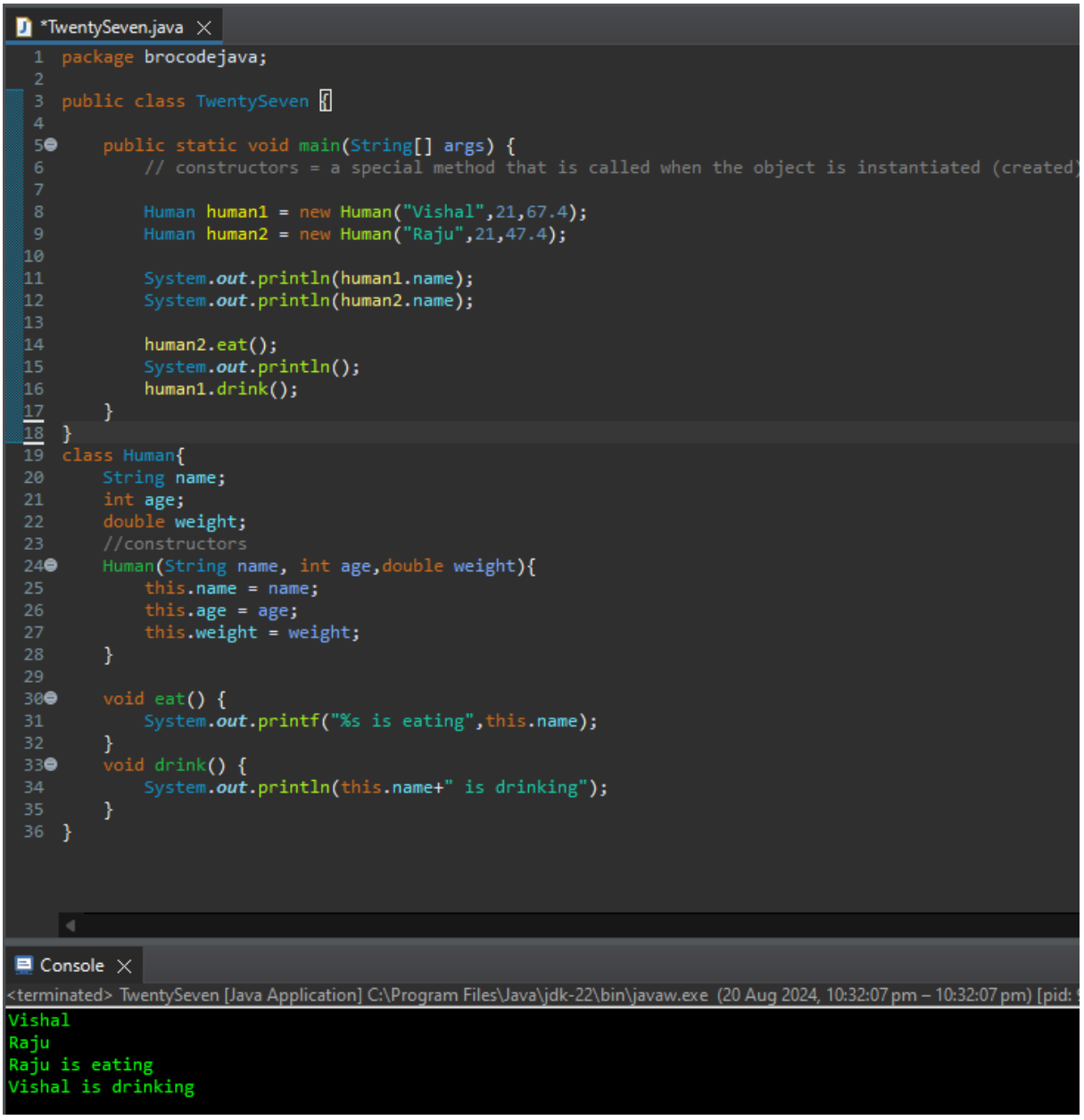
TwentySix.java Car.java

```java
package brocodejava;

public class TwentySix {

    public static void main(String[] args) {

        // object = an instance of a class that may contain attributes and methods
        // example: phone, desk, computer, coffee cup

        Car myCar1 = new Car();
        Car myCar2 = new Car();

        System.out.println(myCar1.model);
        System.out.println(myCar1.make);
        System.out.println();
        System.out.println(myCar2.model);
        System.out.println(myCar2.make);

        myCar1.drive();
        myCar1.brake();

    }

}
```

Car.java

```java
package brocodejava;

public class Car {

    String make = "Suzuki";
    String model = "Ertiga";
    int year = 2020;
    String colour = "Blue";
    double price = 500000.00;


    void drive() {
        System.out.println("You are driving the car");
    }

    void brake() {
        System.out.println("You have pressed the brake!");
    }

}
```

27. Constructors



28. Variable Scope

TwentyEight.java DiceRoller.java

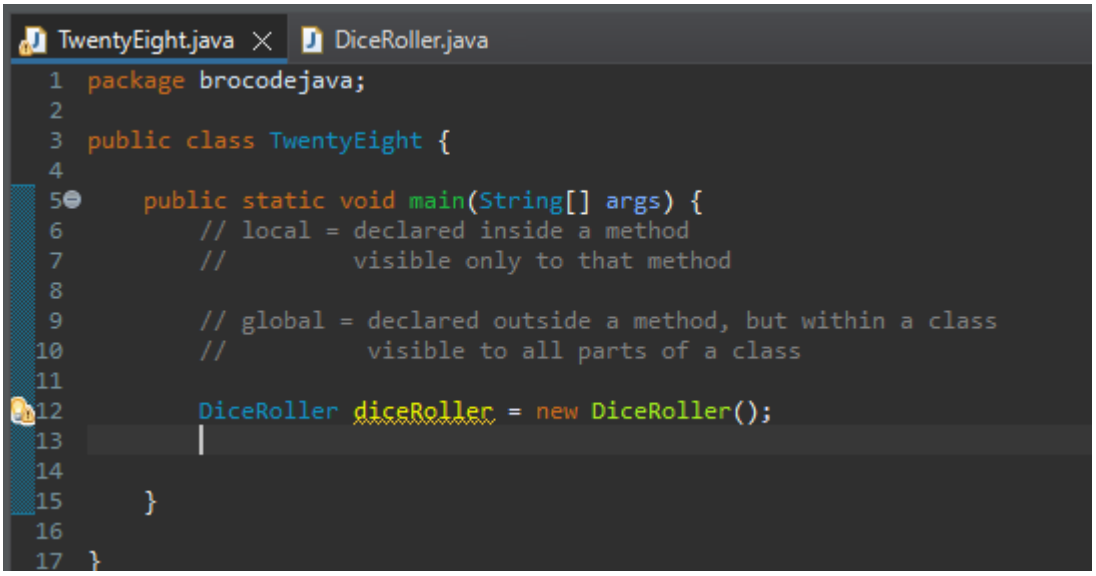
```java
package brocodejava;
import java.util.Random;

public class DiceRoller {

    DiceRoller(){
        Random random = new Random();
        int number = 0;
        roll();
    }
    void roll() {
        number = random.nextInt(6)+1;
        System.out.println(number);
    }


}
```

Random and number only visible to the constructor method

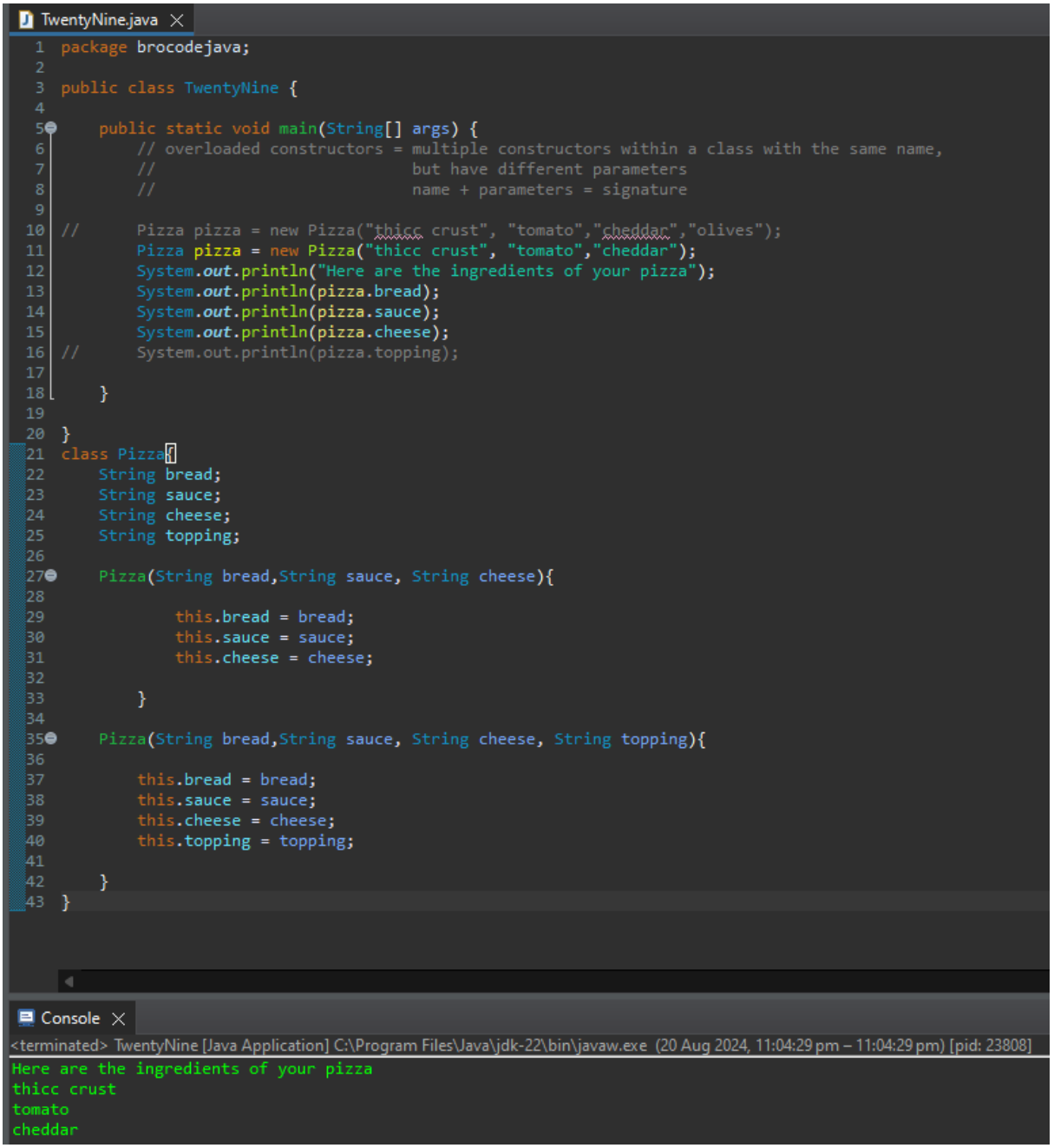
to resolve, either pass them as instance through arguments or declare them global

TwentyEight.java DiceRoller.java

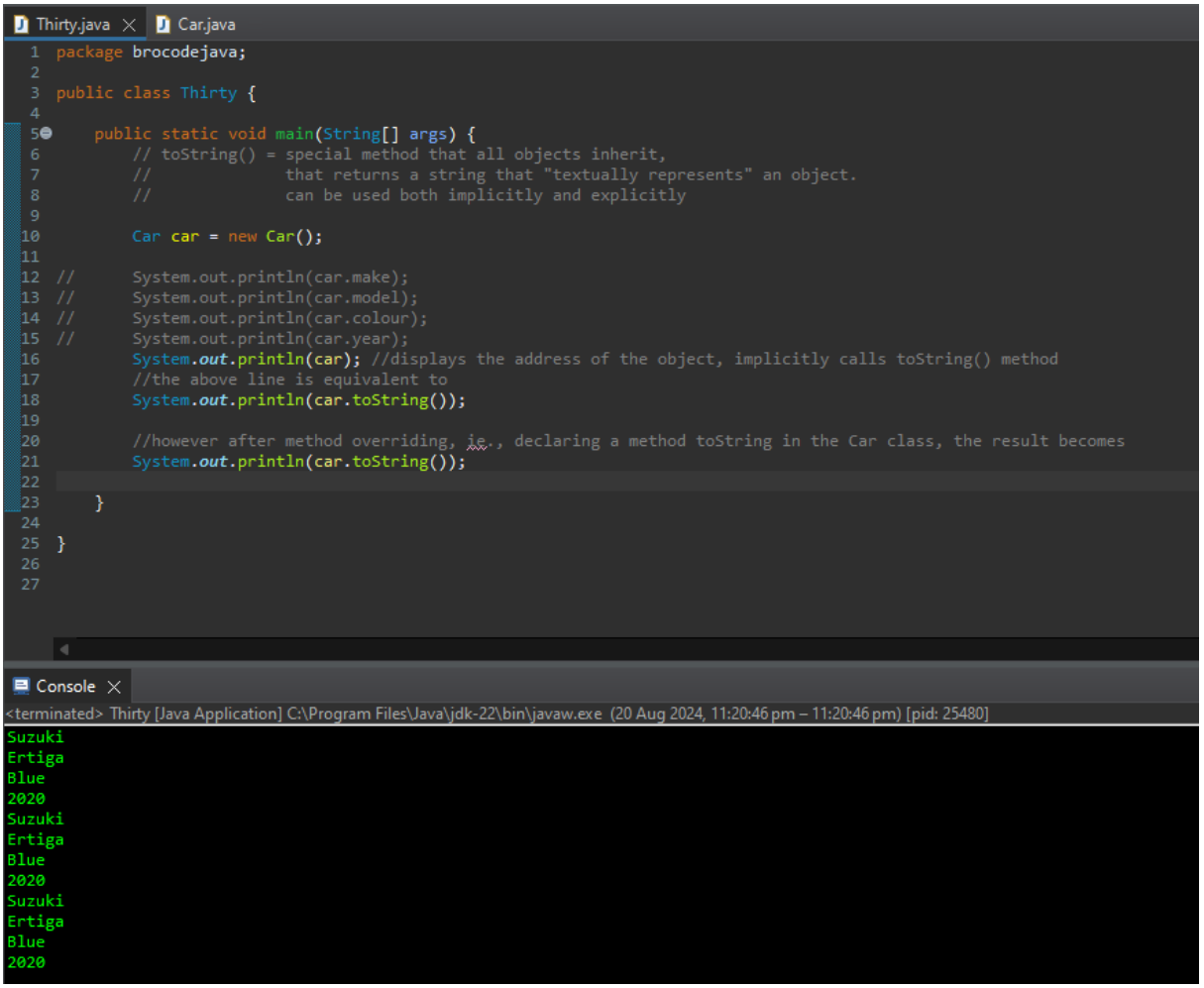
```java
package brocodejava;
import java.util.Random;

public class DiceRoller {

//  DiceRoller(){
//      Random random = new Random();
//      int number = 0;
//      roll(random,number);
//  }
//  void roll(Random random,int number) {
//      number = random.nextInt(6)+1;
//      System.out.println(number);
//  }

    Random random;
    int number = 0;
    DiceRoller(){
        random = new Random();
        roll();
    }
    void roll() {
        number = random.nextInt(6)+1;
        System.out.println(number);
    }


}

```



29. Overloaded Constructors

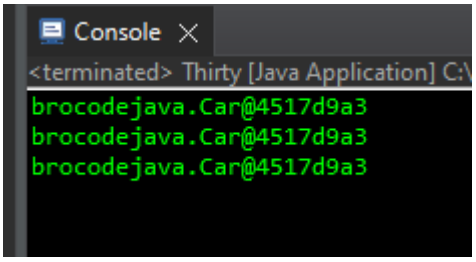


30. toString method


Thirty.java
Car.java
1 package brocodejava;
2
3 public class Thirty {
4
5 public static void main(String[] args) {
6 // toString() = special method that all objects inherit,
7 // that returns a string that "textually represents" an object.
8 // can be used both implicitly and explicitly
9
10 Car car = new Car();
11
12 // System.out.println(car.make);
13 // System.out.println(car.model);
14 // System.out.println(car.colour);
15 // System.out.println(car.year);
16 System.out.println(car); //displays the address of the object, implicitly calls toString() method
17 //the above line is equivalent to
18 System.out.println(car.toString());
19
20 //however after method overriding, ie., declaring a method toString in the Car class, the result becomes
21 System.out.println(car.toString());
22
23 }
24
25 }
26
27
Console
<terminated> Thirty [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (20 Aug 2024, 11:20:46 pm – 11:20:46 pm) [pid: 25480]
Suzuki
Ertiga
Blue
2020
Suzuki
Ertiga
Blue
2020
Suzuki
Ertiga
Blue
2020

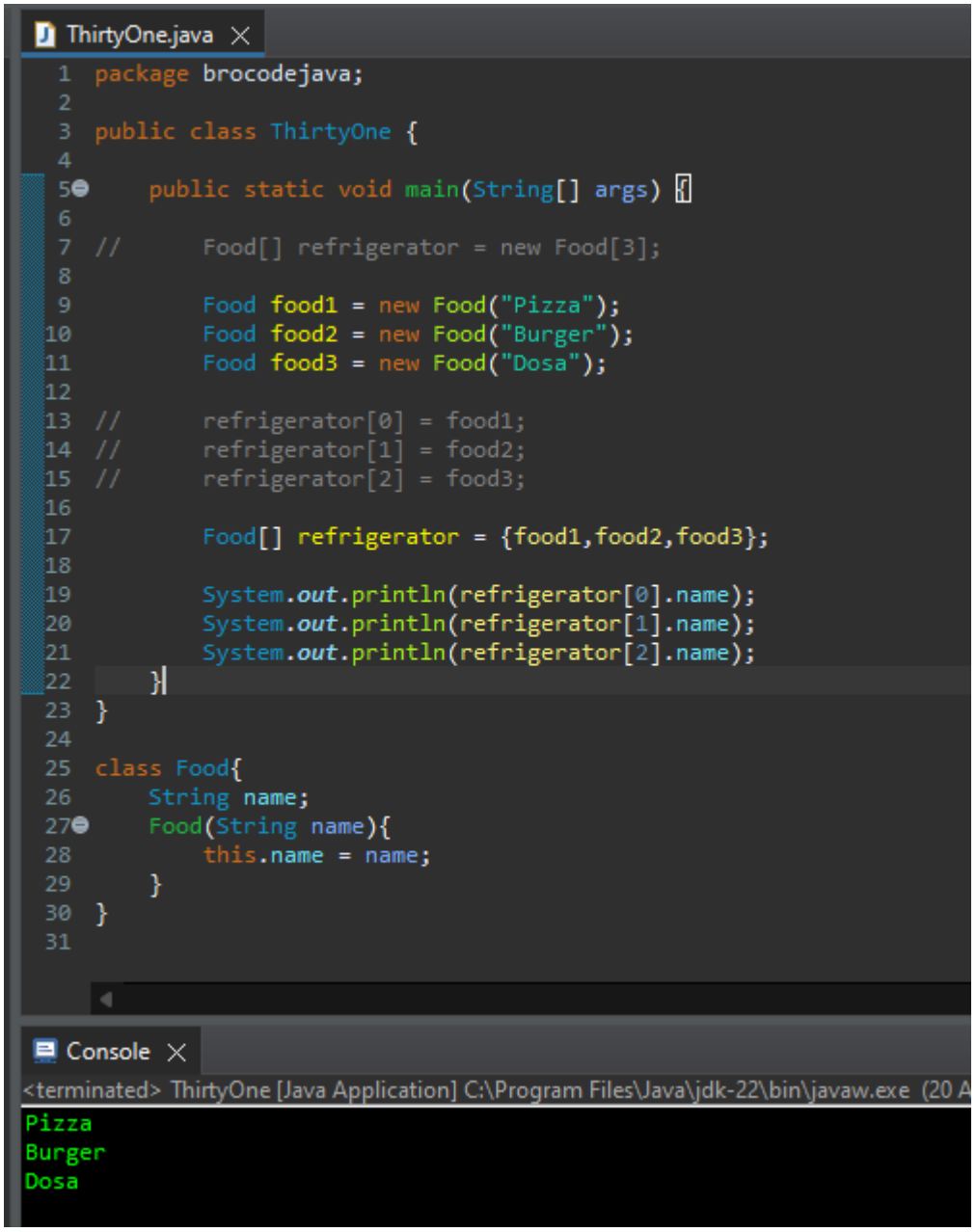
Thirty.java Car.java

```java
package brocodejava;

public class Car {

    String make = "Suzuki";
    String model = "Ertiga";
    int year = 2020;
    String colour = "Blue";
    double price = 500000.00;

    public String toString() {
        String myString = make + "\n" + model + "\n" + colour + "\n" + year;
        return myString;
    }

    void drive() {
        System.out.println("You are driving the car");
    }

    void brake() {
        System.out.println("You have pressed the brake!");
    }

}

```

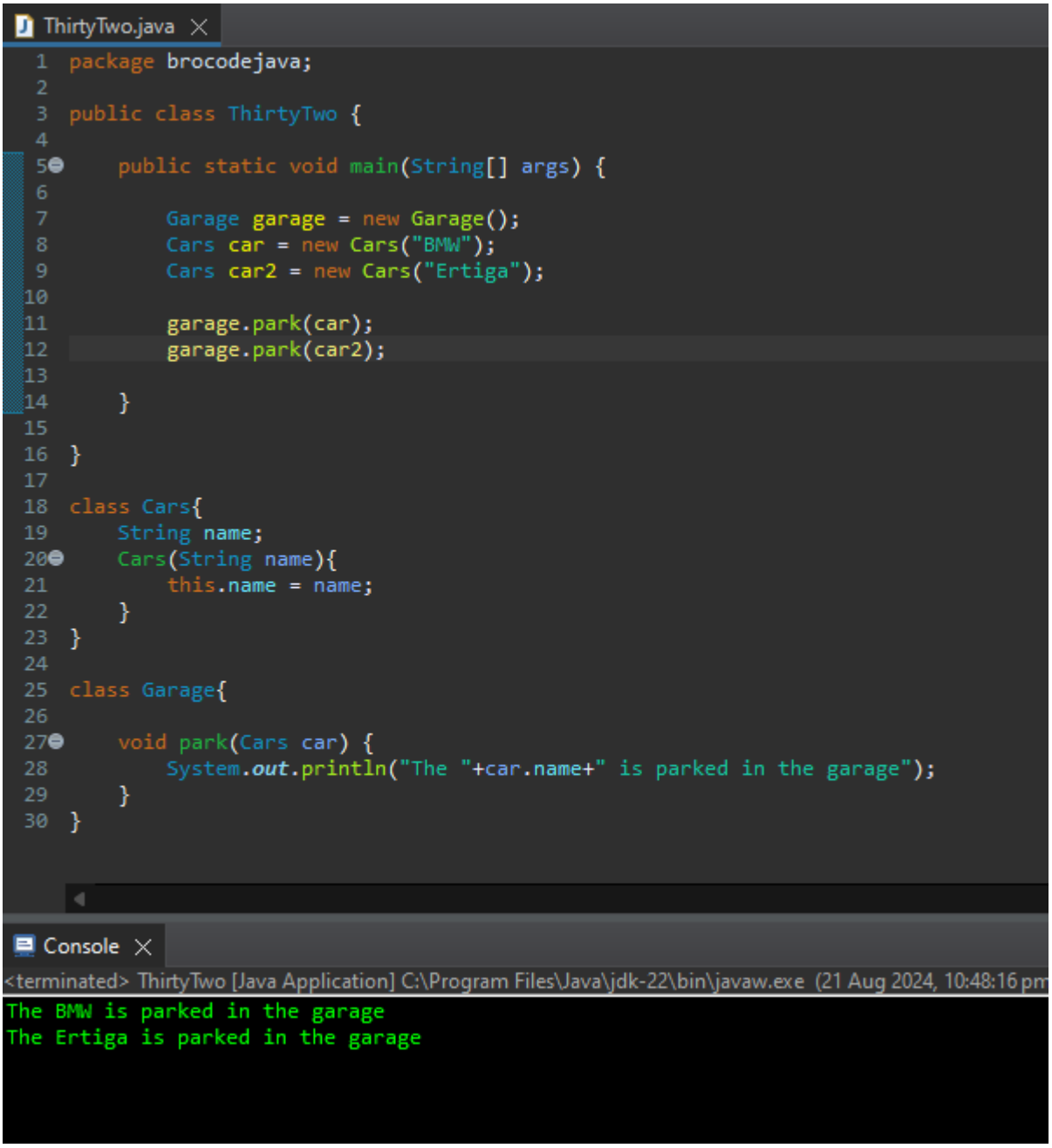


without toString() function in Car class

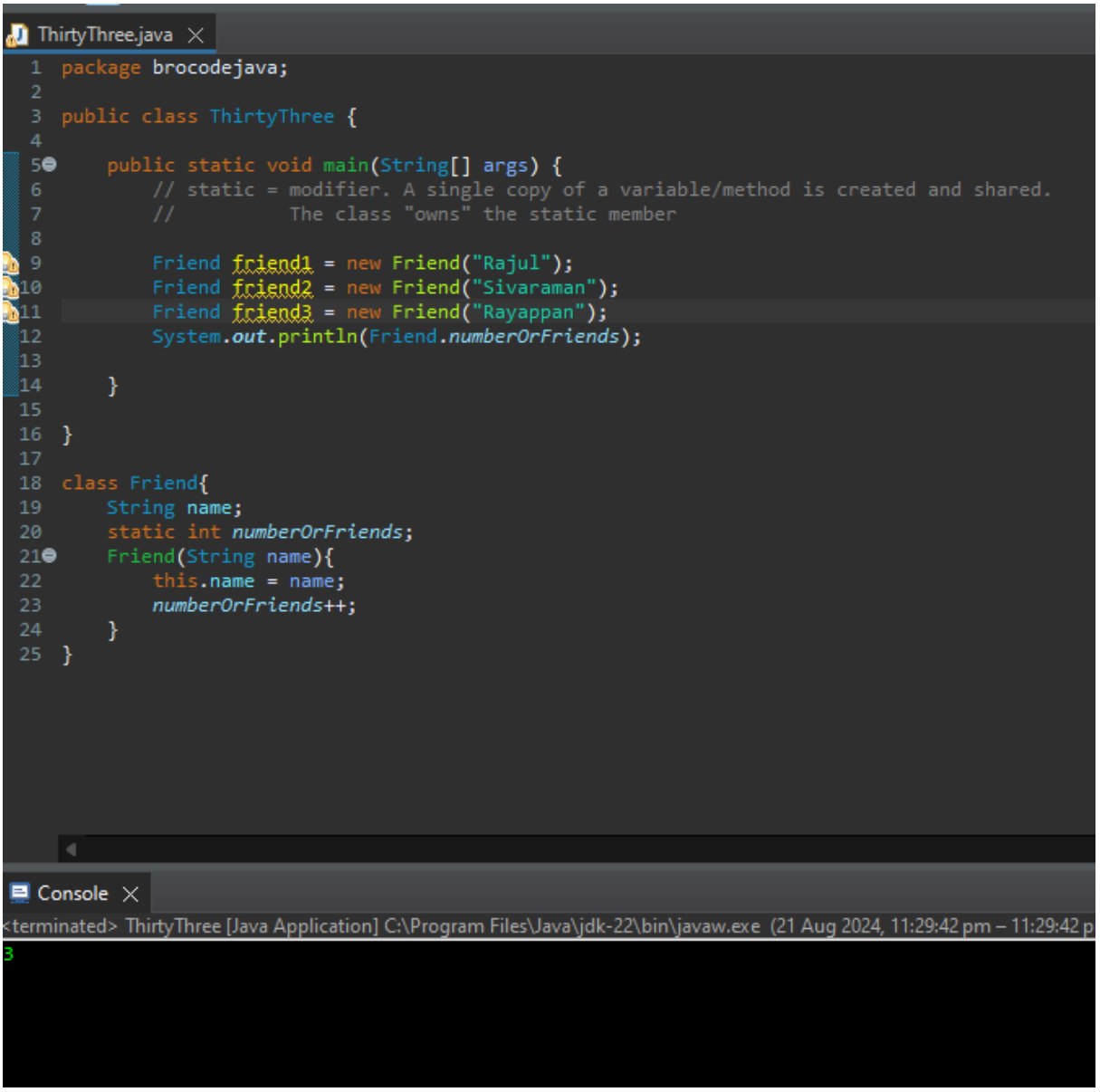
31. Array of Objects



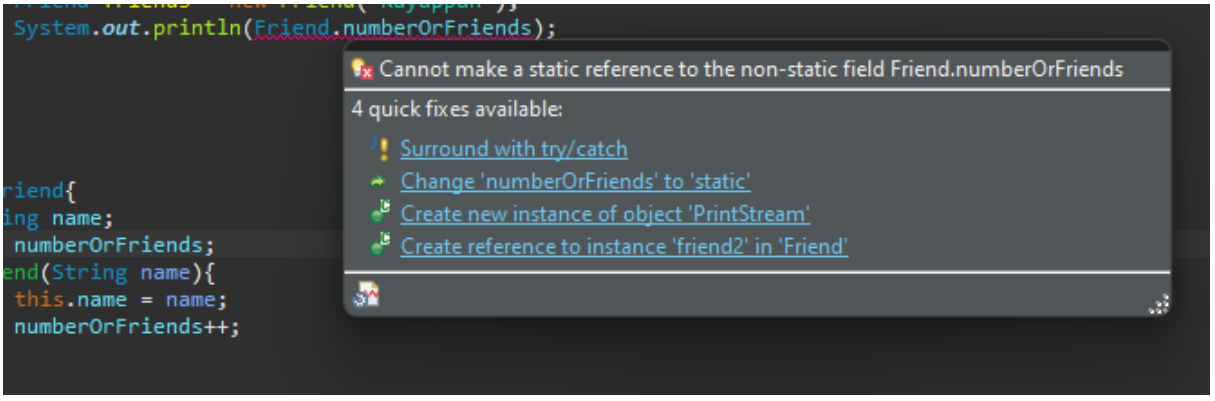
32. Object Passing



33. Static Keyword

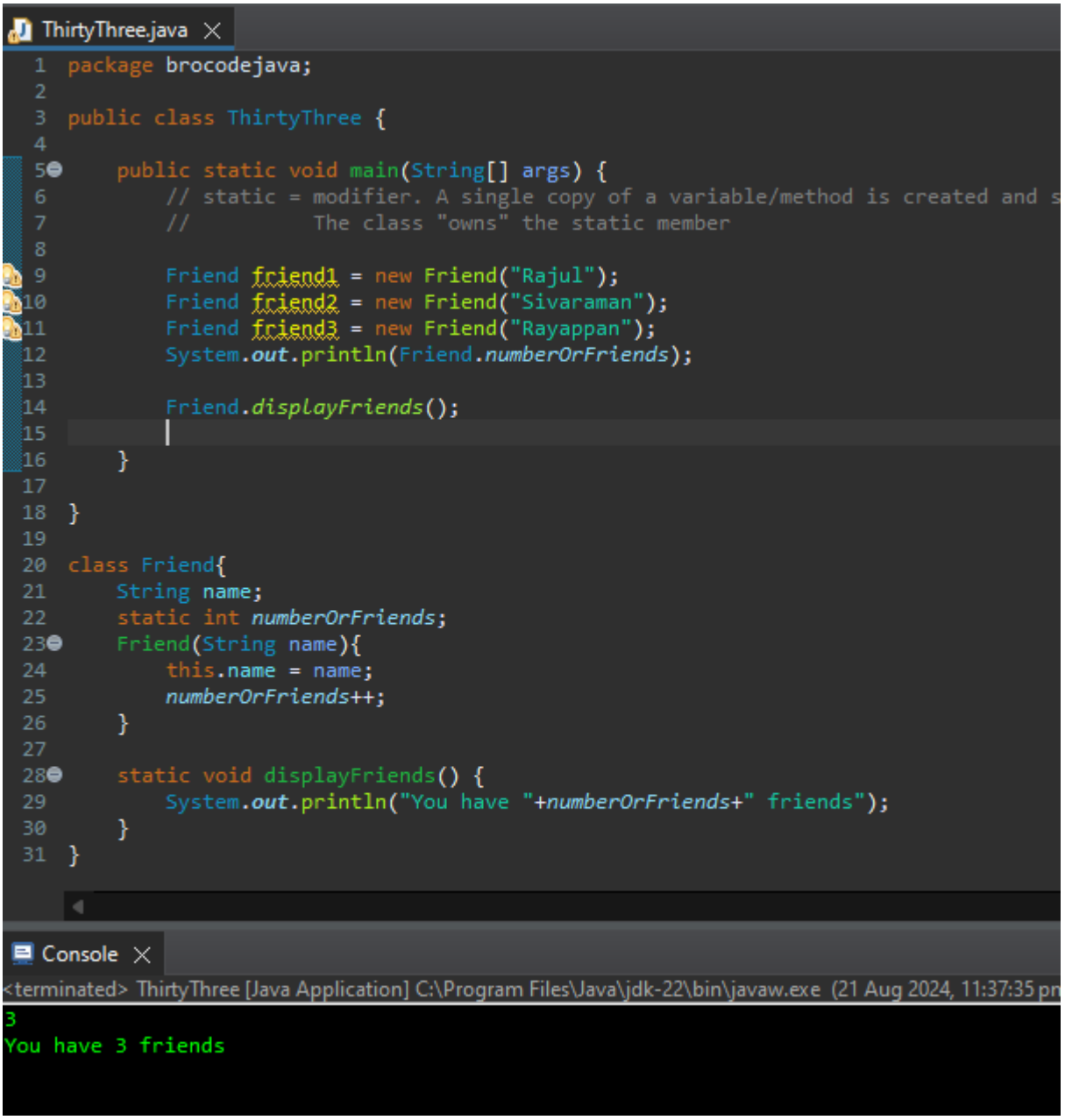


if we remove the static keyword



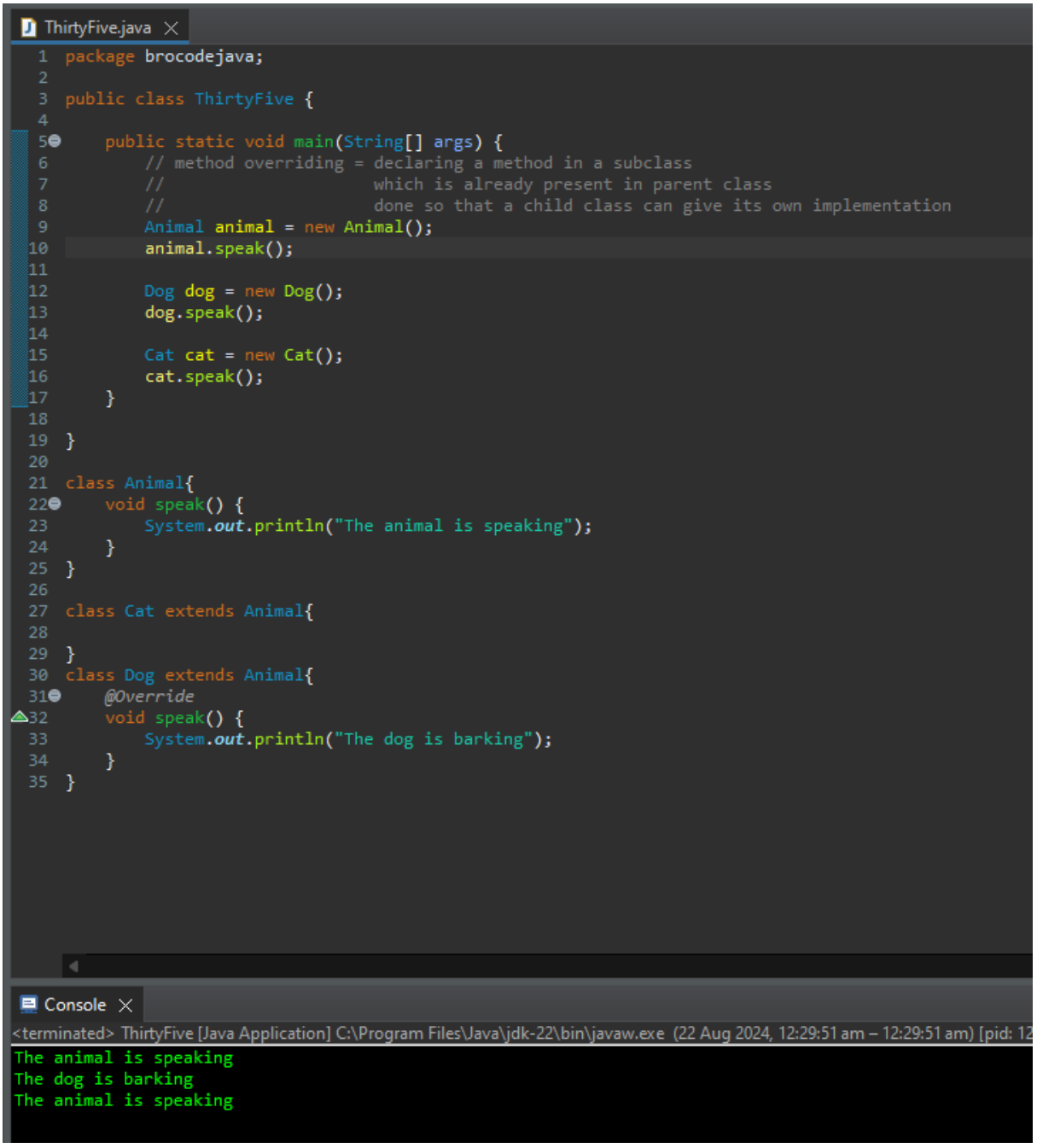
the class owns the singly copy of the static variable

hence every object which uses that class can access it and it will have the same value



34. Inheritance

35. Method Overriding



36. Super Keyword

ThirtySix.java

```java
    public static void main(String[] args) {
        // super = keyword refers to the superclass (parent) of an object.
        //         very similar to the "this" keyword.

        Hero hero1 = new Hero("Batman",42,"$$$");
        Hero hero2 = new Hero("Shaktimaan",75,"infinity");

        System.out.println(hero1.name);
        System.out.println(hero1.age);
        System.out.println(hero1.power);
        System.out.println();
        System.out.println(hero2.toString());
        System.out.println(hero1.toString());
    }
}

class Person{
    String name;
    int age;

    Person(String name,int age){
        this.name = name;
        this.age = age;
    }

    public String toString() { //why are we using public here?
        return this.name + "\n" + this.age + "\n";
    }

}

class Hero extends Person{
    String power;
    Hero(String name, int age, String power){
        super(name,age);
        this.power=power;
    }

    public String toString() { //why are we using public here?
        return super.toString() +  this.power + "\n";
    }
}
```

Console

<terminated> ThirtySix [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (22 Aug 2024, 12:45:40 am -

```
Batman
42
$$$

Shaktimaan
75
infinity

Batman
42
$$$
```

37. Abstraction

ThirtySeven.java

```java
package brocodejava;

public class ThirtySeven {

    public static void main(String[] args) {
        // abstract = abstract classes cannot be instantiated, but they can have a subclass
        //            abstract methods are declared without an implementation
        //            adding abstract keyword adds a level of security
        //Vehicles vehicle = new Vehicles();
        Carzz car = new Carzz();
        car.go();

    }

}

abstract class Vehicles{ // by using abstract keyword, vehicle cannot be declared abstractly and only the child objects can be created.
    abstract void go(); //Abstract methods do not specify a body


}

class Carzz extends Vehicles{
    @Override
    void go() {
        System.out.println("The driver is driving car");
    }
}
```

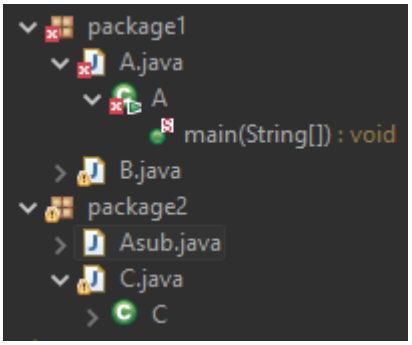
Console

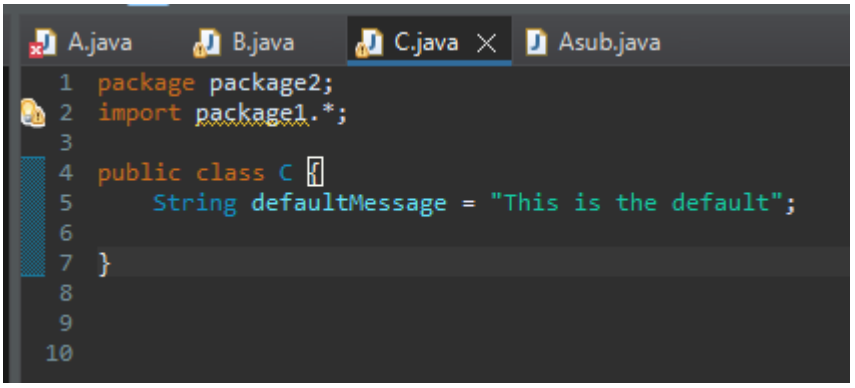
```
<terminated> ThirtySeven [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (22 Aug 2024, 12:54:47 am – 12:54:47 am) [pid: 24332]
The driver is driving car
```

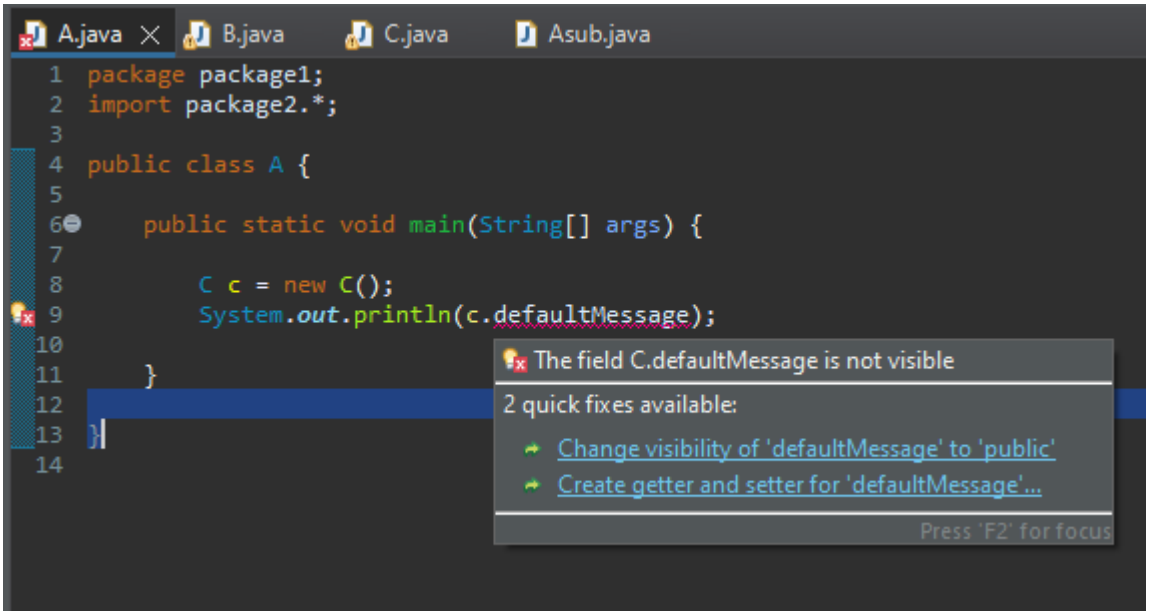
38. Access Modifiers

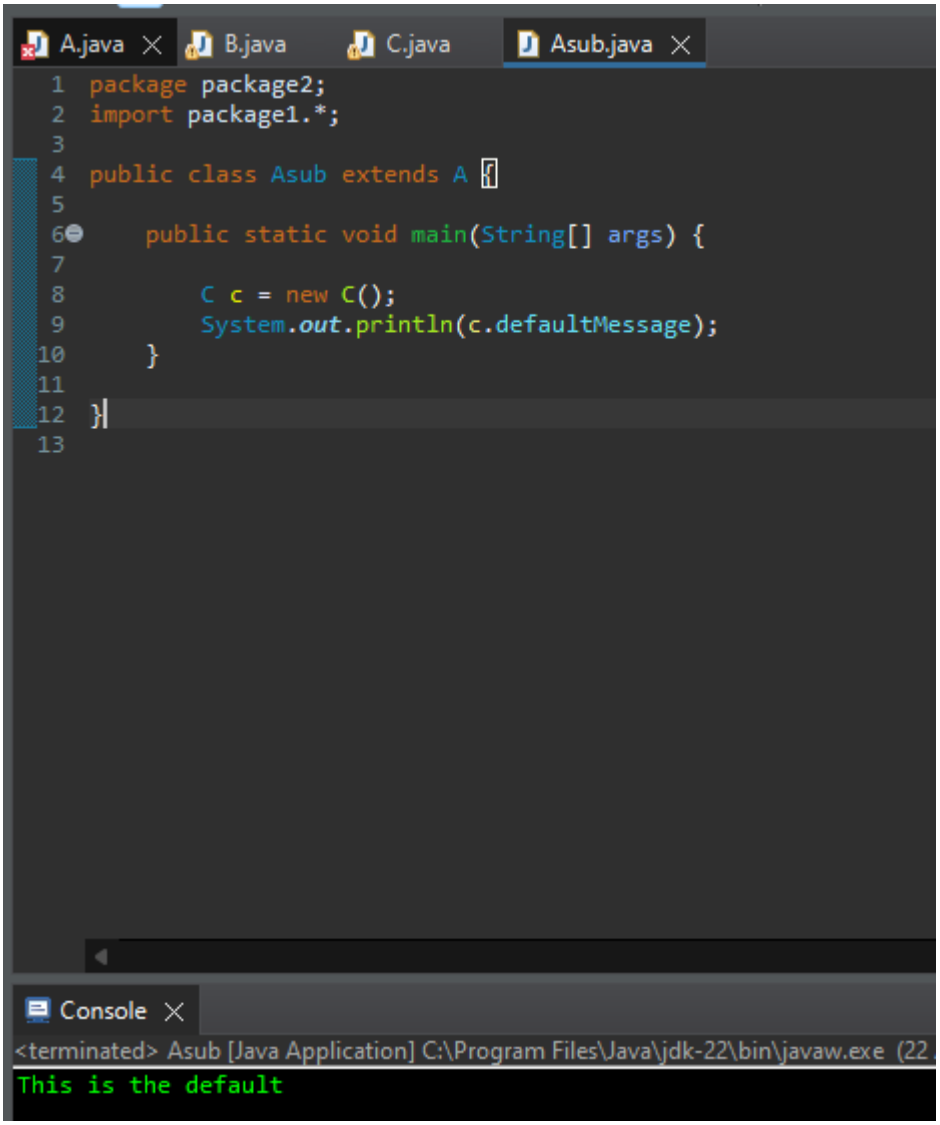
Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

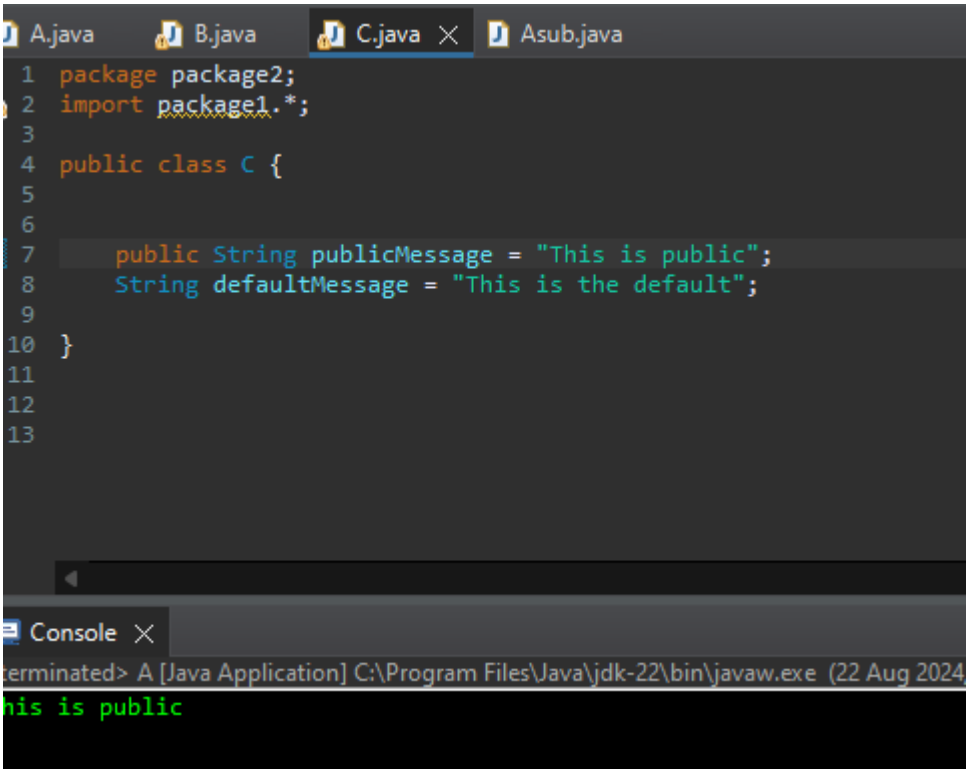

package1
A.java
A
main(String[]) : void
B.java
package2
Asub.java
C.java
C

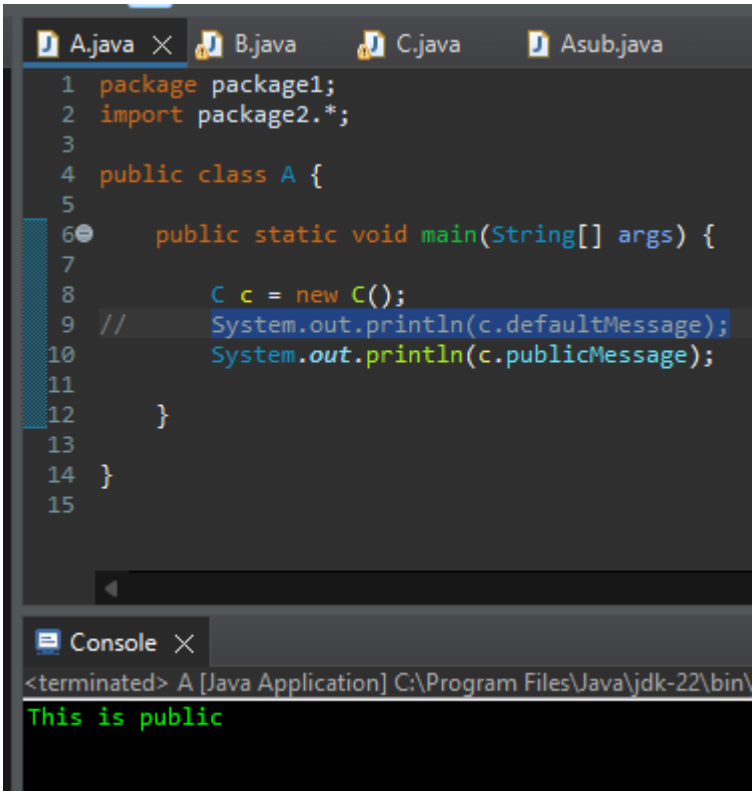

A.java
B.java
C.java
Asub.java
1 package package2;
2 import package1.*;
3
4 public class C {
5 String defaultMessage = "This is the default";
6
7 }
8
9
10


A.java
B.java
C.java
Asub.java
1 package package1;
2 import package2.*;
3
4 public class A {
5
6 public static void main(String[] args) {
7
8 C c = new C();
9 System.out.println(c.defaultMessage);
10
11 }
12
13 }
14
The field C.defaultMessage is not visible
2 quick fixes available:
Change visibility of 'defaultMessage' to 'public'
Create getter and setter for 'defaultMessage'...
Press 'F2' for focus



using public keyword





If we remove the keyword public from the class C, the methods and attributes of that class wont be accessible to classes of other package

protected keyword

A.java B.java C.java Asub.java

```java
package package1;
import package2.*;

public class A {

    protected String protectedMessage = "This is protected";


}

```

A.java B.java C.java Asub.java

```java
package package2;
import package1.*;

public class Asub extends A {

public static void main(String[] args) {

        Asub asub = new Asub();
        System.out.println(asub.protectedMessage);
//      C c = new C();
////        System.out.println(c.defaultMessage);
//      System.out.println(c.publicMessage);

    }

}

```

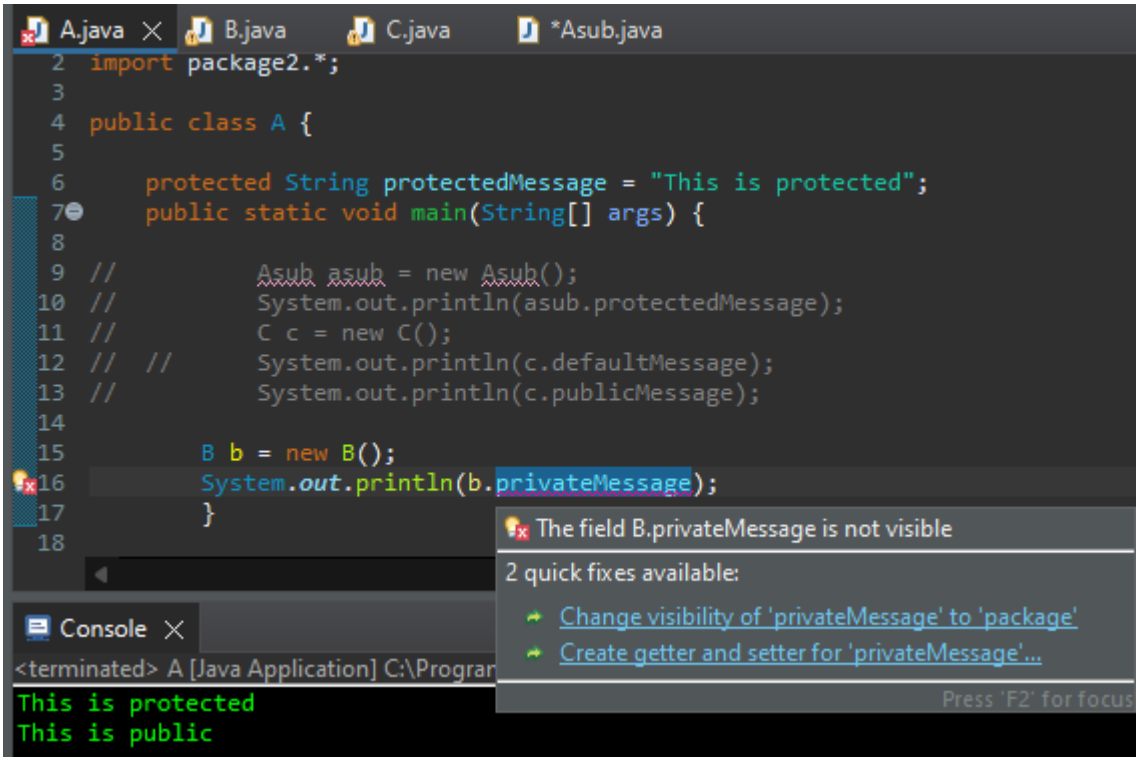
Console

```
<terminated> Asub [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe
This is protected
```

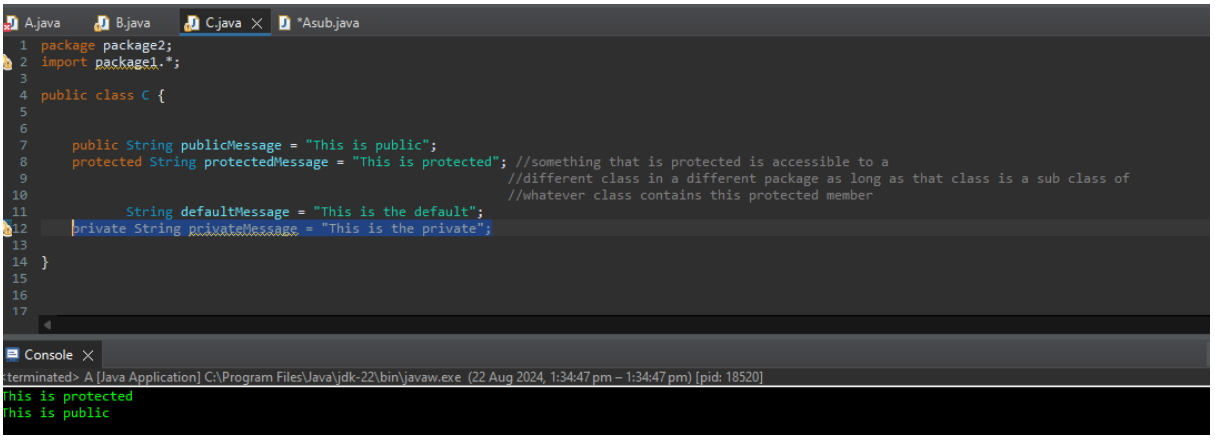
we are able to access protectedMessage from different package as Asub extends A and the A contains protected variable

A.java B.java C.java *Asub.java

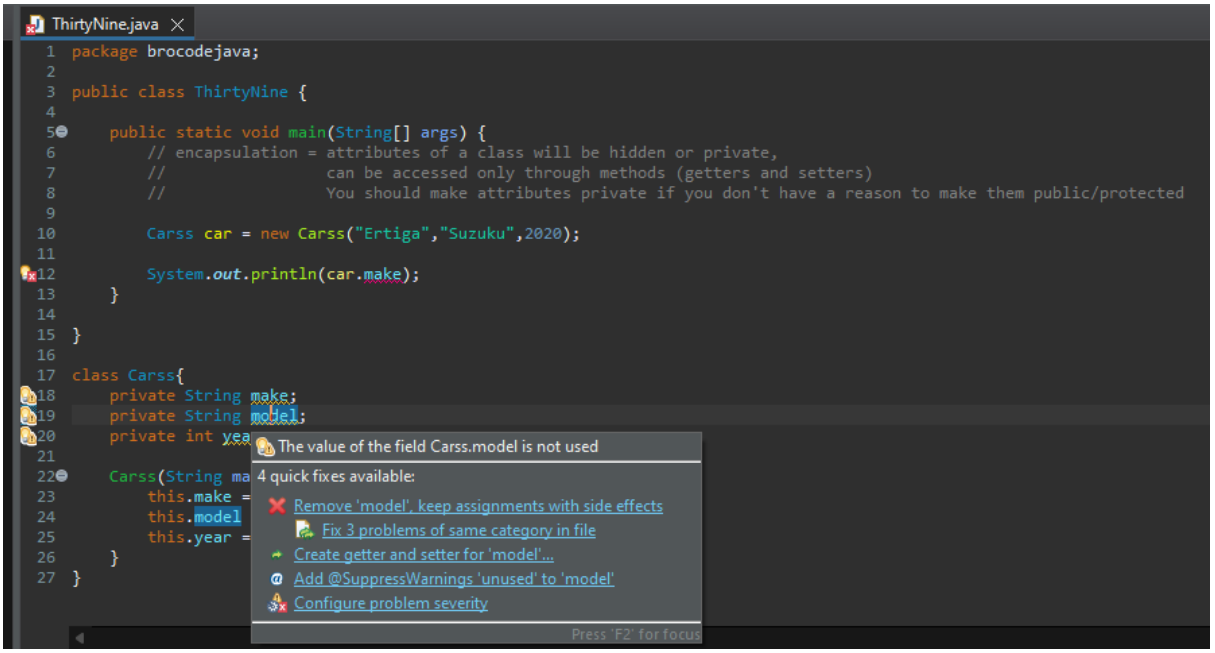
```java
package package1;
import package2.*;

public class B {

    private String privateMessage = "This is the private"; //something that is private is only visible to class that it contains itself

}

```

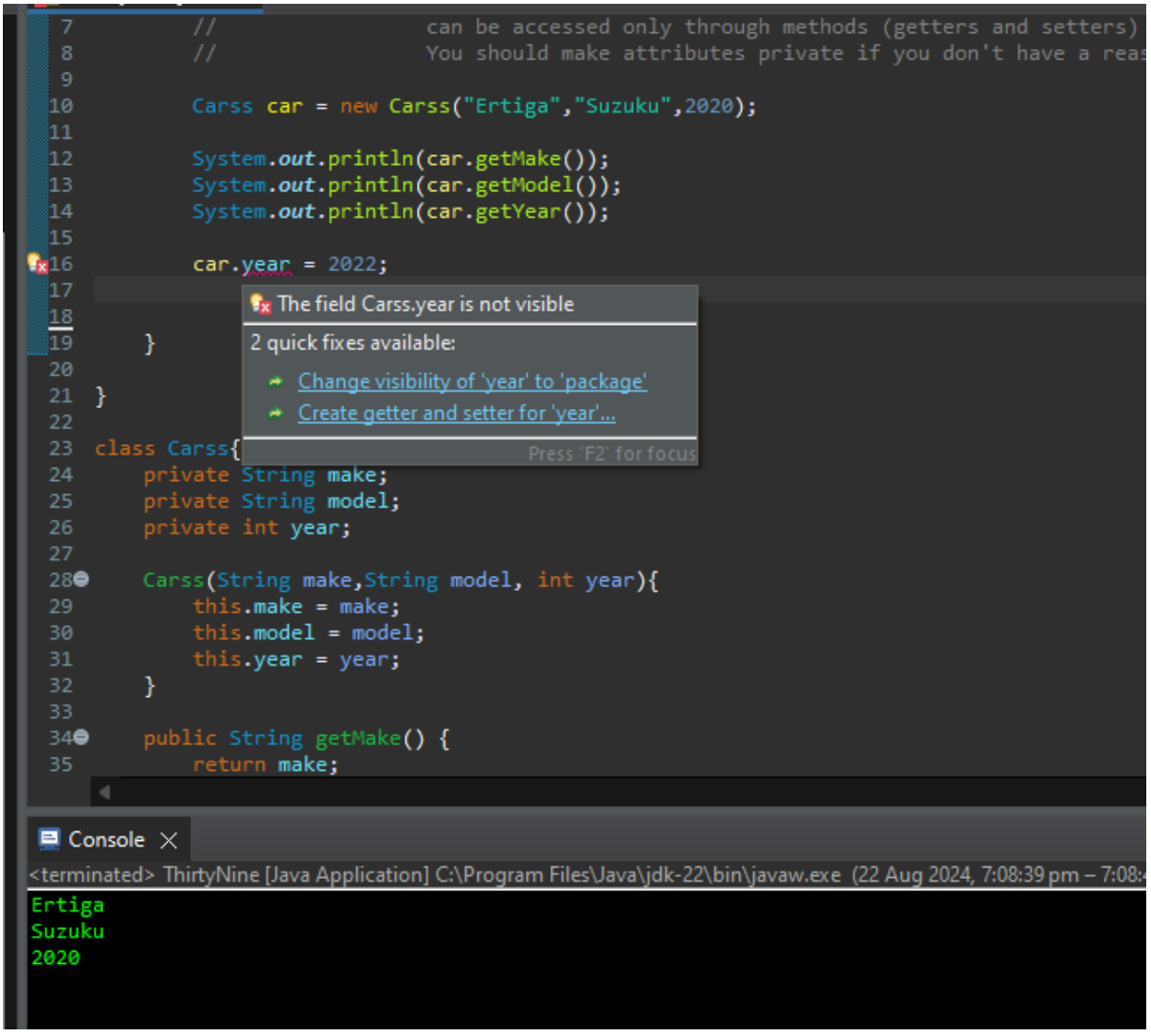


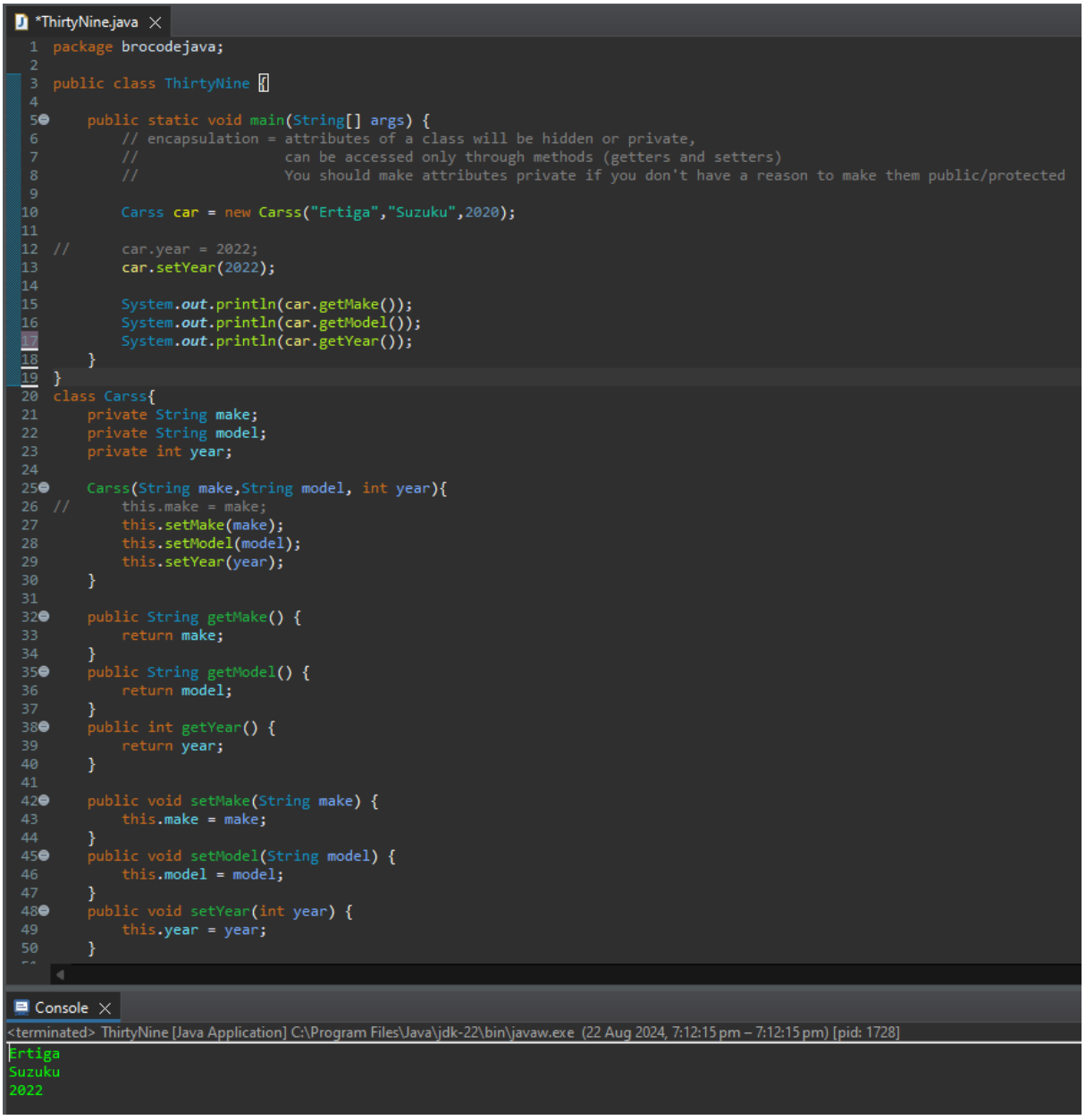
extra:



39. Encapsulation


ThirtyNine.java
package brocodejava;
public class ThirtyNine {
public static void main(String[] args) {
// encapsulation = attributes of a class will be hidden or private,
// can be accessed only through methods (getters and setters)
// You should make attributes private if you don't have a reason to make them public/protected
Carss car = new Carss("Ertiga","Suzuku",2020);
System.out.println(car.make);
}
}
class Carss{
private String make;
private String model;
private int yea
Carss(String ma
this.make =
this.model
this.year =
}
}
The value of the field Carss.model is not used
4 quick fixes available:
Remove 'model', keep assignments with side effects
Fix 3 problems of same category in file
Create getter and setter for 'model'...
Add @SuppressWarnings 'unused' to 'model'
Configure problem severity
Press 'F2' for focus

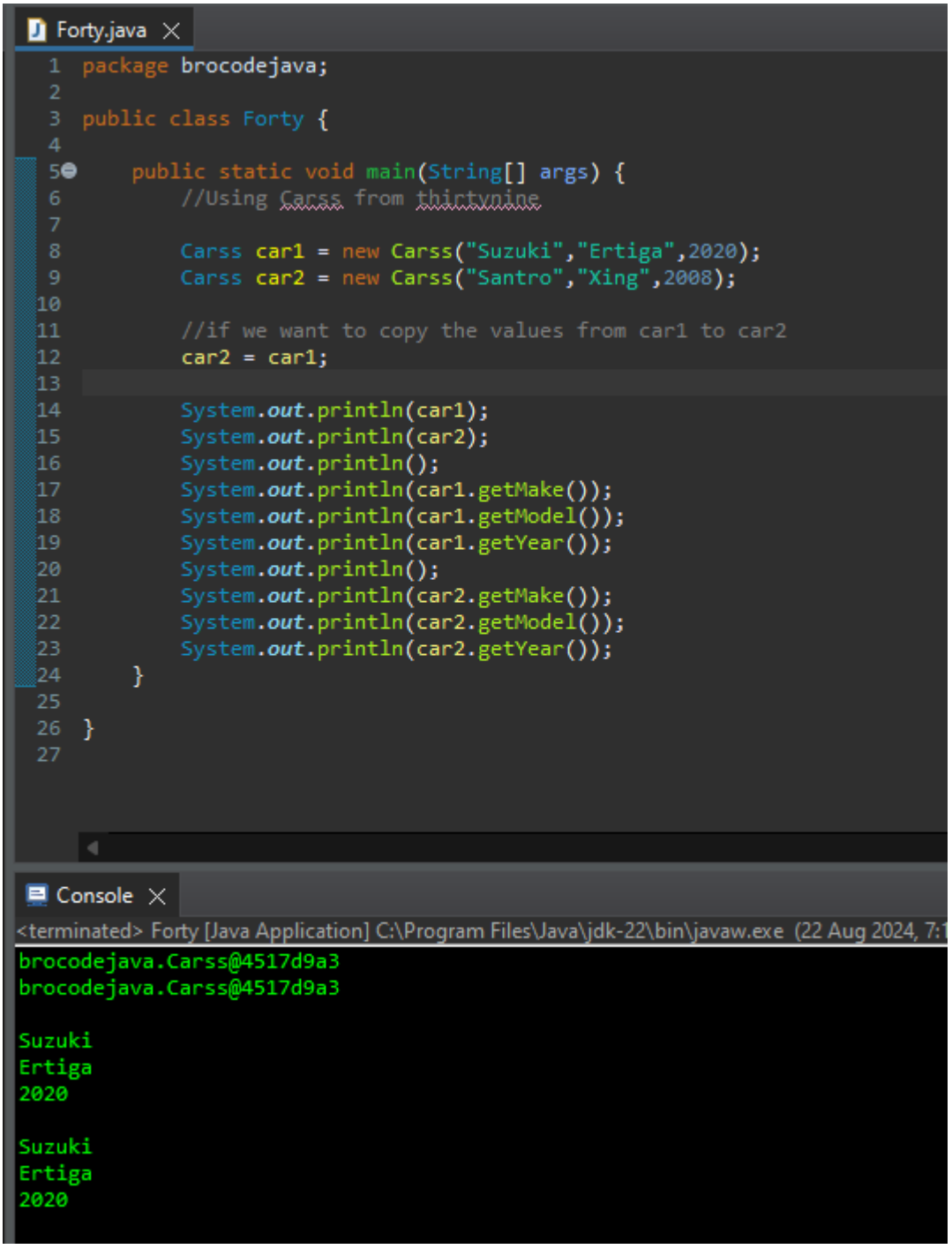

// can be accessed only through methods (getters and setters)
// You should make attributes private if you don't have a reas
Carss car = new Carss("Ertiga","Suzuku",2020);
System.out.println(car.getMake());
System.out.println(car.getModel());
System.out.println(car.getYear());
car.year = 2022;
The field Carss.year is not visible
2 quick fixes available:
Change visibility of 'year' to 'package'
Create getter and setter for 'year'...
Press 'F2' for focus
}
class Carss{
private String make;
private String model;
private int year;
Carss(String make,String model, int year){
this.make = make;
this.model = model;
this.year = year;
}
public String getMake() {
return make;
Console
<terminated> ThirtyNine [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (22 Aug 2024, 7:08:39 pm – 7:08:
Ertiga
Suzuku
2020



Other Classes do not have direct access to the values.

Instead they can use getter and setter methods to access them

40. Copy Objects



both will be referring to the same object, ie,, they point to the same car

Forty.java | ThirtyNine.java

```java
package brocodejava;

public class Forty {

    public static void main(String[] args) {
        //Using Carss from thirtynine

        Carss car1 = new Carss("Suzuki","Ertiga",2020);
        Carss car2 = new Carss("Santro","Xing",2008);

        //if we want to copy the values from car1 to car2
//      car2 = car1;
        car2.copy(car1);

        System.out.println(car1);
        System.out.println(car2);
        System.out.println();
        System.out.println(car1.getMake());
        System.out.println(car1.getModel());
        System.out.println(car1.getYear());
        System.out.println();
        System.out.println(car2.getMake());
        System.out.println(car2.getModel());
        System.out.println(car2.getYear());
    }

}
```

Console

```
<terminated> Forty [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (22 Aug 2024, 7:43:21 pm – 7:43:
brocodejava.Carss@4517d9a3
brocodejava.Carss@372f7a8d

Suzuki
Ertiga
2020

Suzuki
Ertiga
2020
```

```java
        this.year = year;
    }

    public void copy(Carss x) {
        this.setMake(x.getMake());
        this.setModel(x.getModel());
        this.setYear(x.getYear());
    }

}
```

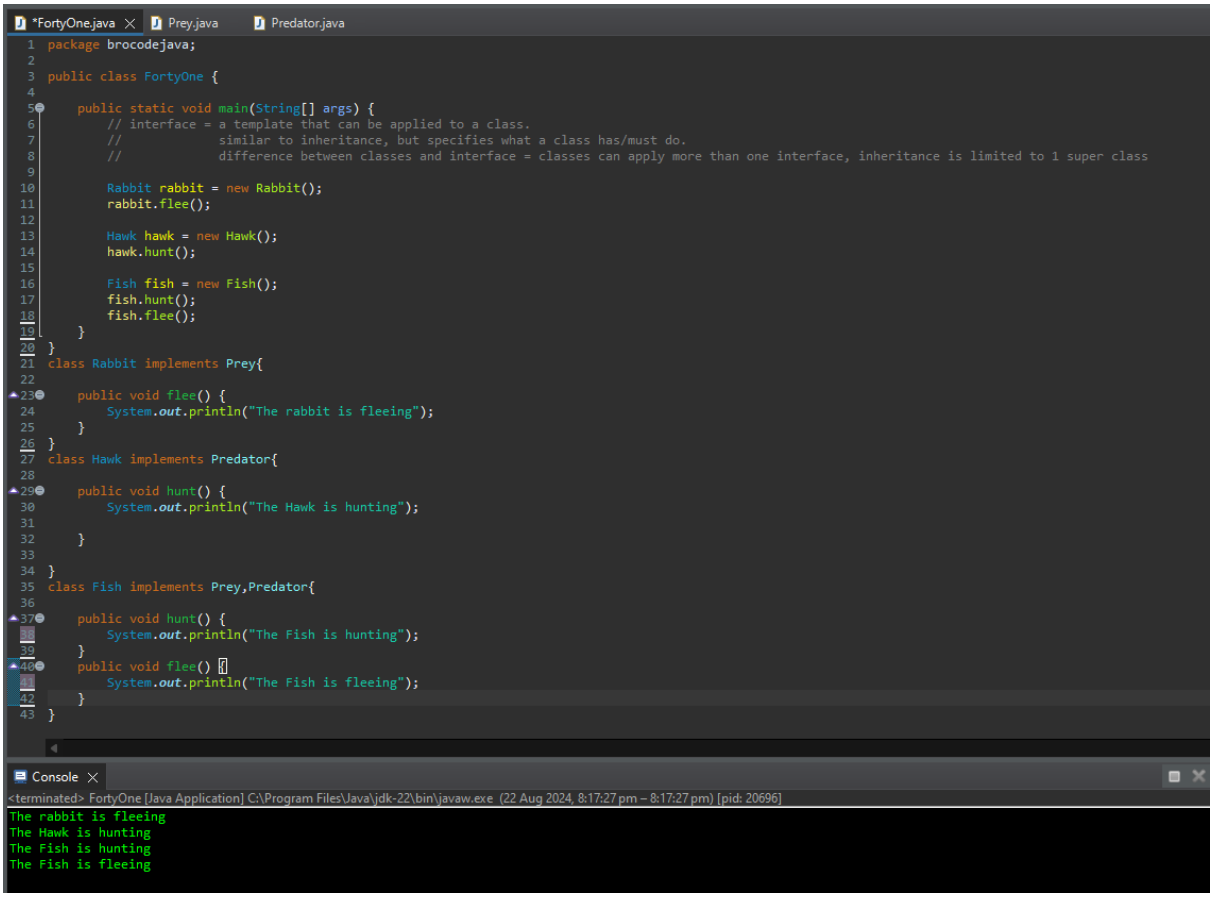
Alternatively, if we want to assign the values of car1 directly to car2

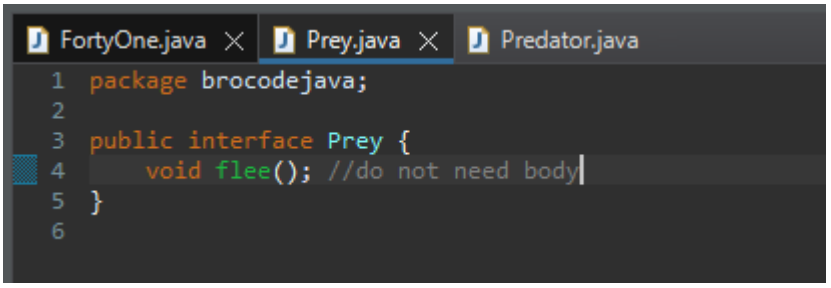
Forty.java | ThirtyNine.java

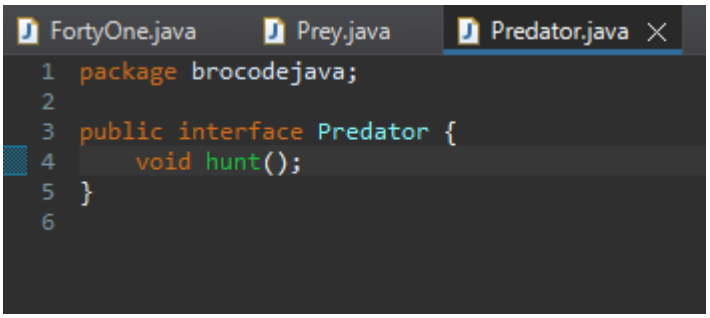
```java
package brocodejava;

public class Forty {

    public static void main(String[] args) {
        //Using Carss from thirtynine

        Carss car1 = new Carss("Suzuki","Ertiga",2020);
//      Carss car2 = new Carss("Santro","Xing",2008);

        //if we want to copy the values from car1 to car2
//      car2 = car1;
//      car2.copy(car1);

        Carss car2 = new Carss(car1);

        System.out.println(car1);
        System.out.println(car2);
        System.out.println();
        System.out.println(car1.getMake());
        System.out.println(car1.getModel());
        System.out.println(car1.getYear());
        System.out.println();
        System.out.println(car2.getMake());
        System.out.println(car2.getModel());
        System.out.println(car2.getYear());
    }

}
```

```java
        this.setYear(year);
    }

    //Overloaded constructors
    Carss(Carss x){
        this.copy(x);
    }
```

41. Interfaces


*FortyOne.java
Prey.java
Predator.java
package brocodejava;
public class FortyOne {
public static void main(String[] args) {
// interface = a template that can be applied to a class.
// similar to inheritance, but specifies what a class has/must do.
// difference between classes and interface = classes can apply more than one interface, inheritance is limited to 1 super class
Rabbit rabbit = new Rabbit();
rabbit.flee();
Hawk hawk = new Hawk();
hawk.hunt();
Fish fish = new Fish();
fish.hunt();
fish.flee();
}
}
class Rabbit implements Prey{
public void flee() {
System.out.println("The rabbit is fleeing");
}
}
class Hawk implements Predator{
public void hunt() {
System.out.println("The Hawk is hunting");
}
}
class Fish implements Prey,Predator{
public void hunt() {
System.out.println("The Fish is hunting");
}
public void flee() {
System.out.println("The Fish is fleeing");
}
}
Console
<terminated> FortyOne [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (22 Aug 2024, 8:17:27 pm – 8:17:27 pm) [pid: 20696]
The rabbit is fleeing
The Hawk is hunting
The Fish is hunting
The Fish is fleeing


FortyOne.java
Prey.java
Predator.java
package brocodejava;
public interface Prey {
void flee(); //do not need body
}


FortyOne.java
Prey.java
Predator.java
package brocodejava;
public interface Predator {
void hunt();
}

FortyOne.java | Prey.java | Predator.java

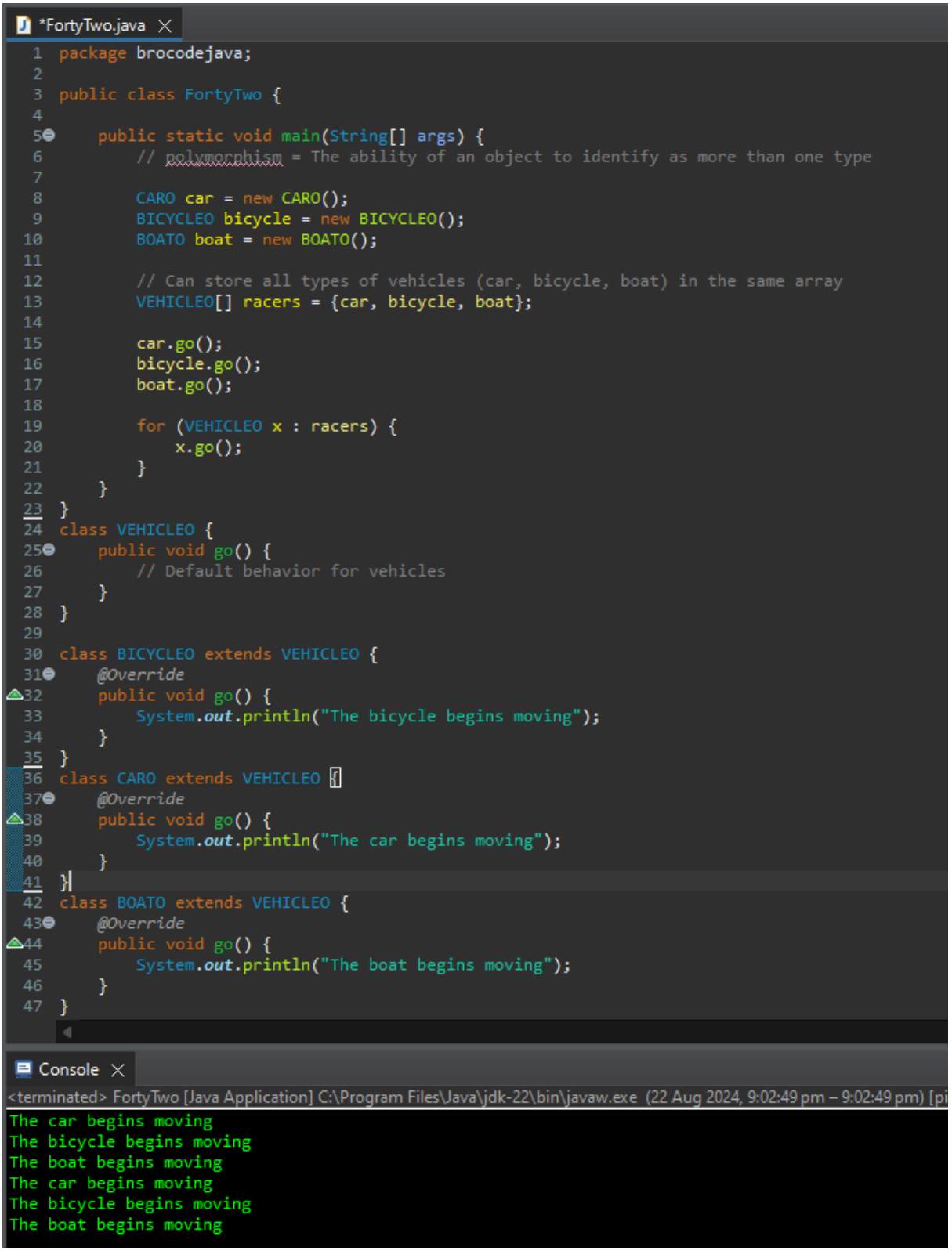
```java
package brocodejava;

public class FortyOne {

    public static void main(String[] args) {
        // interface = a template that can be applied to a class.
        //             similar to inheritance, but specifies what a class has/must do.
        //             difference between classes and interface = classes can apply more than one interface, inheritance is limited to 1 super class

        Rabbit rabbit = new Rabbit();
        rabbit.flee();

        Hawk hawk = new Hawk();
        hawk.hunt();

        Fish fish = new Fish();
        fish.hunt();
        fish.flee();
    }
}
class Rabbit implements Prey{

    public void flee() {
        System.out.println("The rabbit is fleeing");
    }
}
class Hawk implements Predator{

    public void hunt() {
        System.out.println("The Hawk is hunting");

    }

}
class Fish implements Prey,Predator{

    public void hunt() {
        System.out.println("The Fish is hunting");
    }
    public void flee() {
        System.out.println("The Fish is fleeing");
    }
}
```

Console

<terminated> FortyOne [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (22 Aug 2024, 8:17:27 pm – 8:17:27 pm) [pid: 20696]

```
The rabbit is fleeing
The Hawk is hunting
The Fish is hunting
The Fish is fleeing
```

42. Polymorphism



43. Dynamic Polymorphism

FortyThree.java

```java
package brocodejava;
import java.util.*;
public class FortyThree {

    public static void main(String[] args) {
        //polymorphism = many forms
        // dynamic = after compilation, during runtime
        Scanner scanner = new Scanner(System.in);
        Animal animal;

        System.out.println("What animal do u want?");
        String choice = scanner.next();
        System.out.println(choice);
        if (choice.equals("dog")) {  //using == will check for the memory values also hence would return false
            animal = new Dog();
            animal.speak();
        }
        else if(choice.equals("cat")) { //therefore using .equals() is suitable in this case as it checks for the values only
            animal = new Cat();
            animal.speak();
        }
        else {
            animal = new Animal();
            animal.speak();
        }

    }

}
```

Console

```
<terminated> FortyThree [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (24 Aug 2024, 11:48:07 am – 11:48:42 am) [pid: 19832]
What animal do u want?
cat
cat
The cat says meow
```

44. Exception Handling

FortyFour.java

```java
package brocodejava;
import java.util.Scanner;
public class FortyFour {

    public static void main(String[] args) {
        // exception = an event that occurs during the execution of a program that disrupts the normal flow of instructions
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter a whole number to divide");
        int x = scanner.nextInt();

        System.out.println("Enter a whole number to divide by: ");
        int y = scanner.nextInt();

        int z = x/y;
        System.out.println("result: "+z );

    }

}
```

Console

```
<terminated> FortyFour [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (24 Aug 2024, 11:51:34 am – 11:51:37 am) [pid: 19756]
Enter a whole number to divide
5
Enter a whole number to divide by:
0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at brocodejava.FortyFour.main(FortyFour.java:15)
```

*FortyFour.java

```java
package brocodejava;
import java.util.InputMismatchException;
import java.util.Scanner;
public class FortyFour {

    public static void main(String[] args) {
        // exception = an event that occurs during the execution of a program that disrupts the normal flow of instructions
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.println("Enter a whole number to divide");
            int x = scanner.nextInt();

            System.out.println("Enter a whole number to divide by: ");
            int y = scanner.nextInt();

            int z = x/y;
            System.out.println("result: "+z );
        }
        catch(ArithmeticException e) {
            System.out.println("You cannot divide by 0 rahul");
        }
        catch(InputMismatchException e) {
            System.out.println("Enter number da tpmf");
        }
        catch(Exception e) {
            System.out.println("Something went wrong");
        }
        finally {
            System.out.println("This will always execute");
            scanner.close();
        }
    }
}
```

Console

```
<terminated> FortyFour [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (24 Aug 2024, 12:29:49 pm – 12:29:51 pm) [pid: 30764]
Enter a whole number to divide
10
Enter a whole number to divide by:
0
You cannot divide by 0 rahul
This will always execute
```

FortyFour.java

```java
package brocodejava;
import java.util.InputMismatchException;
import java.util.Scanner;
public class FortyFour {

    public static void main(String[] args) {
        // exception = an event that occurs during the execution of a program that disrupts the normal flow of instructions
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.println("Enter a whole number to divide");
            int x = scanner.nextInt();

            System.out.println("Enter a whole number to divide by: ");
            int y = scanner.nextInt();

            int z = x/y;
            System.out.println("result: "+z );
        }
        catch(ArithmeticException e) {
            System.out.println("You cannot divide by 0 rahul");
        }
        catch(InputMismatchException e) {
            System.out.println("Enter number da tpmf");
        }
        catch(Exception e) {
            System.out.println("Something went wrong");
        }
        finally {
            System.out.println("This will always execute");
            scanner.close();
        }
    }
}
```

Console

```
<terminated> FortyFour [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (24 Aug 2024, 12:30:16 pm – 12:30:19 pm) [pid: 30676]
Enter a whole number to divide
10
Enter a whole number to divide by:
pizza
Enter number da tpmf
This will always execute
```

45. File Class

FortyFive.java

```java
package brocodejava;
import java.io.File;
public class FortyFive {

    public static void main(String[] args) {
        // file = abstract representation of file and directory pathnames

        File file = new File("C:\\Users\\visha\\eclipse-workspace\\brocodejava\\src\\brocodejava\\secret_message.txt");
        if (file.exists()) {
            System.out.println("That file exists");
            System.out.println(file.getPath());
            System.out.println(file.getAbsolutePath());
            System.out.println(file.isFile());
            //file.delete();
        }
        else {
            System.out.println("That file doesnt exist");
        }

    }

}
```

Console

```
<terminated> FortyFive [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (24 Aug 2024, 12:59:00 pm – 12:59:00 pm) [pid: 12032]
That file exists
C:\Users\visha\eclipse-workspace\brocodejava\src\brocodejava\secret_message.txt
C:\Users\visha\eclipse-workspace\brocodejava\src\brocodejava\secret_message.txt
true
```

46. File Writer

FortySix.java

```java
package brocodejava;
import java.io.FileWriter;
import java.io.IOException;
public class FortySix {

    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("C:\\Users\\visha\\eclipse-workspace\\brocodejava\\src\\brocodejava\\secret_message.txt");
            writer.write("This is Secret \nSecret can be Scret ");
            writer.append("\n(Visynthesis)");
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }


    }

}

```

secret_message.txt

```
This is Secret 
Secret can be Scret 
(Visynthesis)
```

47. File Reader

*FortySeven.java

```java
package brocodejava;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class FortySeven {

    public static void main(String[] args) {
        // FileReader = read the contents of a file as a stream of characters. One by one
        //              read() returns an int value which contains the byte value
        //              when read() returns -1, there is no more data to be read

        try {
            FileReader reader = new FileReader("C:\\Users\\visha\\eclipse-workspace\\brocodejava\\src\\brocodejava\\secret_message.txt");
            int data = reader.read();
            while (data != -1) {
                System.out.print((char)data);
                data = reader.read();
            }
            reader.close();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

Console

```
<terminated> FortySeven [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (24 Aug 2024, 1:33:56 pm – 1:33:56 pm) [pid: 2536]
This is Secret
Secret can be Scret
(Visynthesis)
```

48-76 SKIPPED FOR NOW
TO visit: Serialization

77. Threads

In Java, the `InterruptedException` is a checked exception that is thrown when a thread that is sleeping, waiting, or otherwise occupied is interrupted. It is used to handle situations where a thread is waiting or sleeping and another thread wants to interrupt it and make it stop waiting or sleeping.

Common Scenarios Where `InterruptedException` Is Thrown:

1. **Sleeping**: When a thread is put to sleep using `Thread.sleep()`, it can be interrupted by another thread. This will throw an `InterruptedException`.

```java
try {
    Thread.sleep(1000);  // Sleep for 1 second
} catch (InterruptedException e) {
    // Handle the interruption
    System.out.println("Thread was interrupted");
}
```

2. **Waiting**: When a thread is waiting on an object using methods like `Object.wait()` or `Thread.join()`, it can be interrupted.

```java
synchronized (someObject) {
    try {
        someObject.wait();  // Wait until notified
    } catch (InterruptedException e) {
        // Handle the interruption
        System.out.println("Thread was interrupted while waiting");
    }
}
```

3. **Joining**: When a thread is waiting for another thread to finish using `Thread.join()`, it can be interrupted.

java Copy code

```java
Thread t = new Thread(() -> {
    // Thread logic
});
t.start();
try {
    t.join();  // Wait for t to finish
} catch (InterruptedException e) {
    // Handle the interruption
    System.out.println("Thread was interrupted while waiting for another thread to fin
}
```

Why Use `throws InterruptedException`:

- **Declaration Requirement**: If your method calls a method that can throw `InterruptedException` (like `Thread.sleep()`), you either need to handle the exception with a try-catch block or declare that your method can throw it using `throws InterruptedException`.
- **Propagating the Exception**: By using `throws InterruptedException`, you allow the calling method to decide how to handle the interruption. This is useful if your method is part of a larger thread management system.

java Copy code

```java
public void someMethod() throws InterruptedException {
    Thread.sleep(1000);  // Could throw InterruptedException
}
```

Thread = a thread of execution in a program (kind of like a virtual cpu)
A JVM allows an application to have multiple threads running concurrently
Each thread can execute parts of your code in parallel with the main thread
Each thread has a priority
Threads with higher priority are executed in preference compared to threads with lower priority

```
    The Java Virtual Machine continues to execute threads
until either of the following occurs
```

1. The exit method of class Runtime has been called
2. All user threads have died

When a JVM starts up, there is a thread which calls the main method
This thread is called "main"

Daemon thread is a low priority thread that runs in background to perform tasks such as garbage collection
JVM terminates itself when all user threads (non-daemon threads) finish their execution

SeventySeven.java

```java
package brocodejava;

public class SeventySeven {

    public static void main(String[] args) throws InterruptedException {

        System.out.println(Thread.activeCount());
        Thread.currentThread().setName("Z3R3F");
        System.out.println(Thread.currentThread().getName());
        Thread.currentThread().setPriority(10); //Highest Priority
        System.out.println(Thread.currentThread().getPriority());  //Higher the number, higher the priority

        //to check if the current thread is alive
        System.out.println(Thread.currentThread().isAlive());

        for (int i=3;i>0;i--) {
            System.out.println(i);
            Thread.sleep(1000);
        }
        System.out.println("You are done!");
    }

}
```

Console

```
<terminated> SeventySeven [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (24 Aug 2024, 1:53:09 pm – 1:53:12 pm) [pid: 10316]
1
Z3R3F
10
true
3
2
1
You are done!
```

*SeventySeven.java × MyThread.java

```java
public class SeventySeven {

    public static void main(String[] args) throws InterruptedException {

        System.out.println(Thread.activeCount());
        Thread.currentThread().setName("Z3R3F");
        System.out.println(Thread.currentThread().getName());
        Thread.currentThread().setPriority(10); //Highest Priority
        System.out.println(Thread.currentThread().getPriority());  //Higher the number, higher the priority
        //to check if the current thread is alive
        System.out.println(Thread.currentThread().isAlive());
        for (int i=3;i>0;i--) {
            System.out.println(i);
            Thread.sleep(1000);
        }
        System.out.println("You are done!");

        MyThread thread2 = new MyThread();

        System.out.println("DAEMON?");
        System.out.println(thread2.isDaemon());
        System.out.println("DAEMON?");
        thread2.setDaemon(true);
        System.out.println(thread2.isDaemon());

        thread2.start();
        System.out.println(thread2.isAlive());
        thread2.setName("Threadex");
        System.out.println(thread2.getName());

        System.out.println(thread2.getPriority()); //inherits the priority of the main thread
        thread2.setPriority(9);
        System.out.println(thread2.getPriority());

        System.out.println(Thread.activeCount());
    }
```

eventySeven.java × MyThread.java ×

```java
package brocodejava;

public class MyThread extends Thread {

    //overriding the method run in thread class
    @Override
    public void run() {
        if(this.isDaemon()) {
            System.out.println("This is a Daemon thread that is running");
        }
        else {
            System.out.println("This is a user thread that is running");
        }
        System.out.println("This Thread is Running");
    }


}
```

```
<terminated> SeventySeven [Java Application] C:\Progra
1
Z3R3F
10
true
3
2
1
You are done!
DAEMON?
false
DAEMON?
true
true
Threadex
10
9
This is a Daemon thread that is running
This Thread is Running
2
```

78. Multi-Threading

SeventyEight.java | MyThread2.java | MyRunnable.java

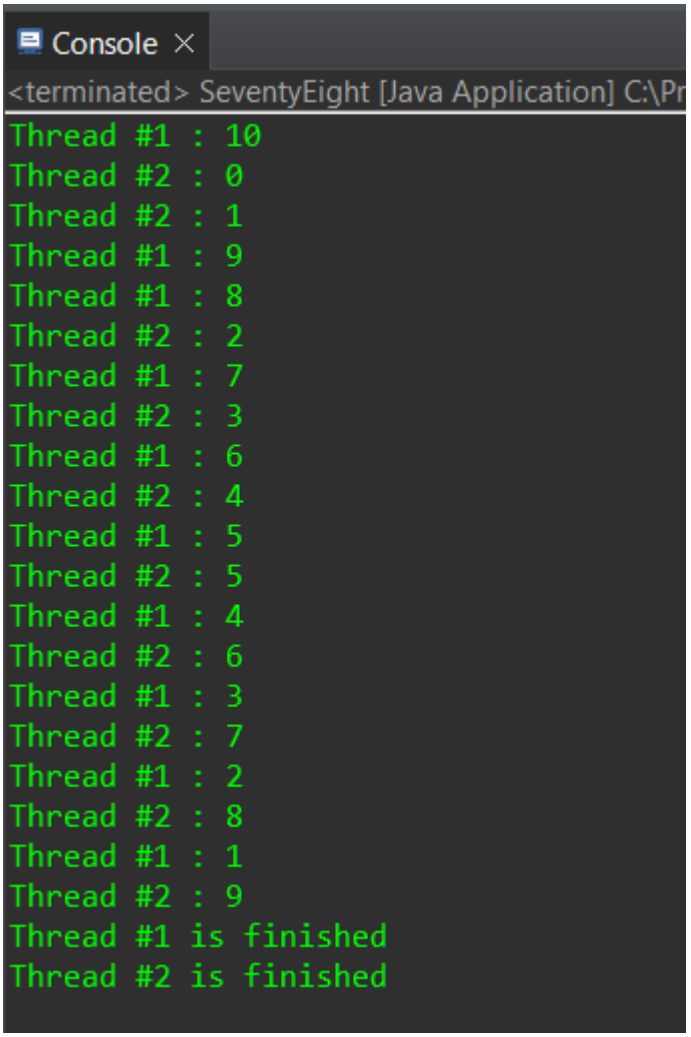
```java
package brocodejava;

public class SeventyEight {

    public static void main(String[] args) {
        // multithreading = process of executing multiple threads simultaneously
        //                  Helps maximum utilization of CPU
        //                  Threads are independent, they don't affect the execution of other threads
        //                  An exception in one thread will not interrupt other threads
        //                  useful for serving multiple clients, multiplayer games, or other mutually independent tasks

        //1
        //create a subclass of the thread class, create a class and make sure it extends thread class,
        //and then you have access to the run method
        MyThread2 thread1 = new MyThread2();

        //2
        //other way to create a thread -> using subclass or class thats implementing runnable interface,
        //create instance and pass it as an argument to the thread class
        MyRunnable runnable1 = new MyRunnable();
        Thread thread2 = new Thread(runnable1);

        thread1.start();
        thread2.start();
```

SeventyEight.java | MyThread2.java | MyRunnable.java

```java
package brocodejava;

public class MyThread2 extends Thread {

    public void run() {

        for(int i=10;i>0;i--) {
            System.out.println("Thread #1 : "+i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        System.out.println("Thread #1 is finished");
    }
}
```

SeventyEight.java | MyThread2.java | MyRunnable.java

```java
package brocodejava;

public class MyRunnable implements Runnable{
    public void run() {

        for(int i=0;i<10;i++) {
            System.out.println("Thread #2 : "+i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        System.out.println("Thread #2 is finished");

    }

}
```



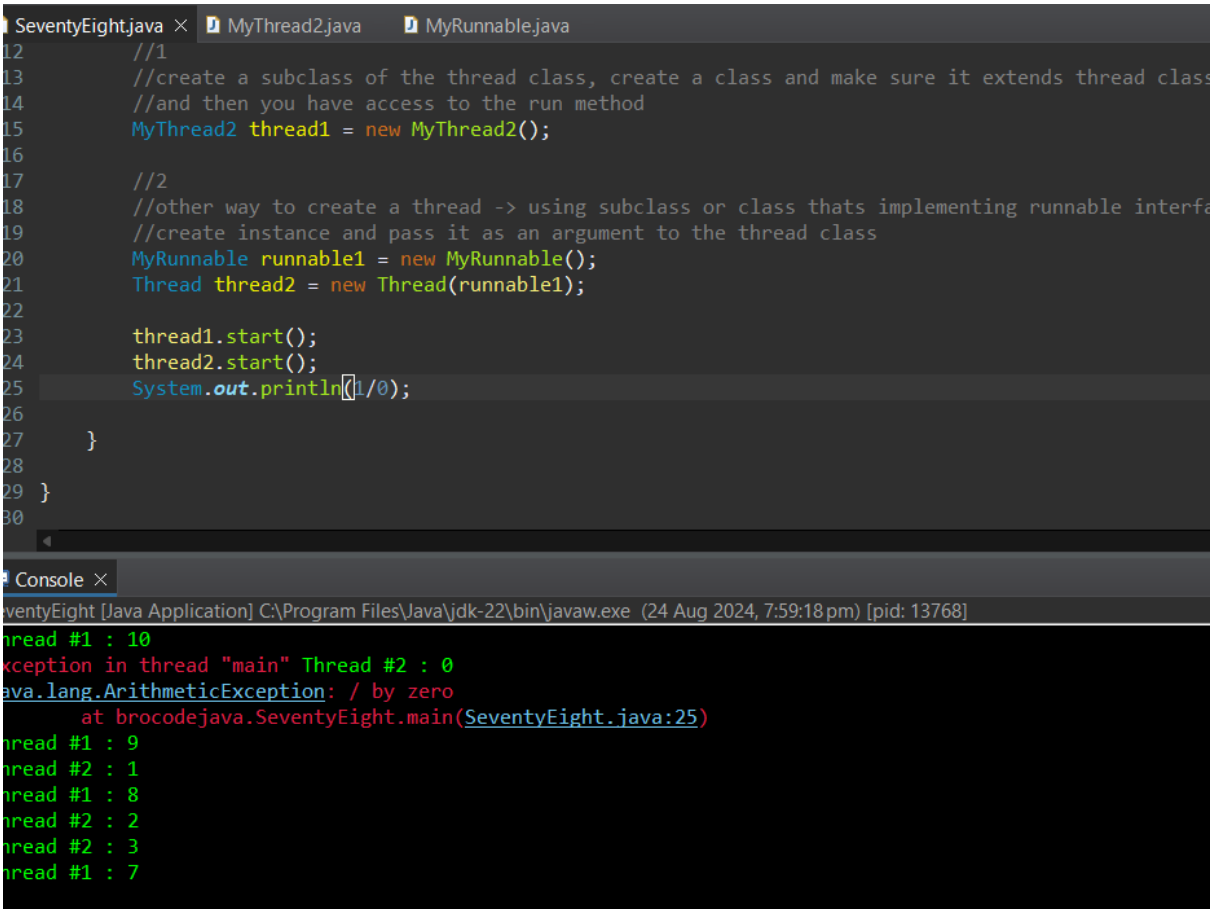
Intentionally causing one thread to stop executing

SeventyEight.java | MyThread2.java | MyRunnable.java

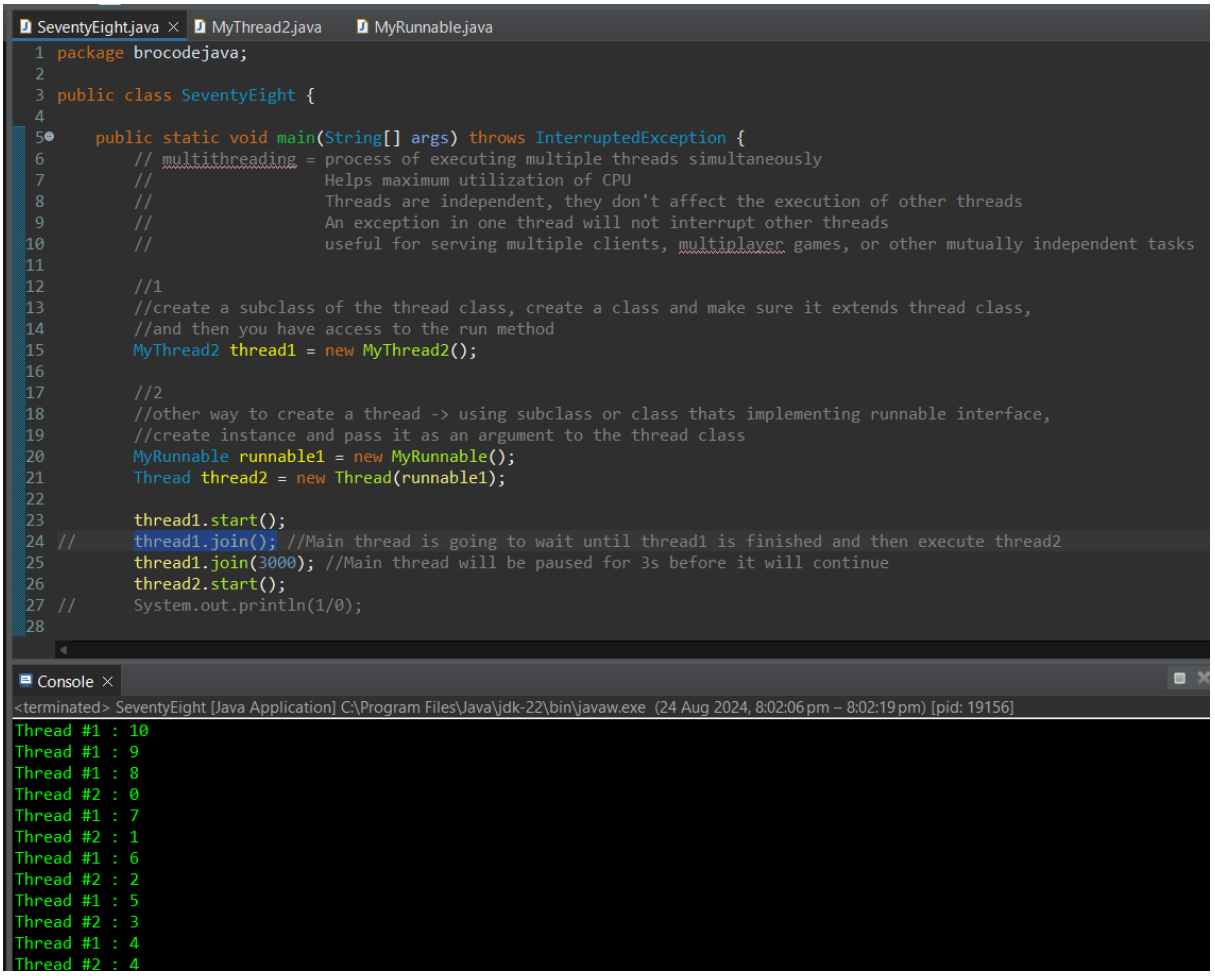
```java
package brocodejava;

public class MyThread2 extends Thread {

    public void run() {

        for(int i=10;i>0;i--) {
            System.out.println("Thread #1 : "+i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            System.out.println(1/0);
        }
        System.out.println("Thread #1 is finished");
    }
}
```

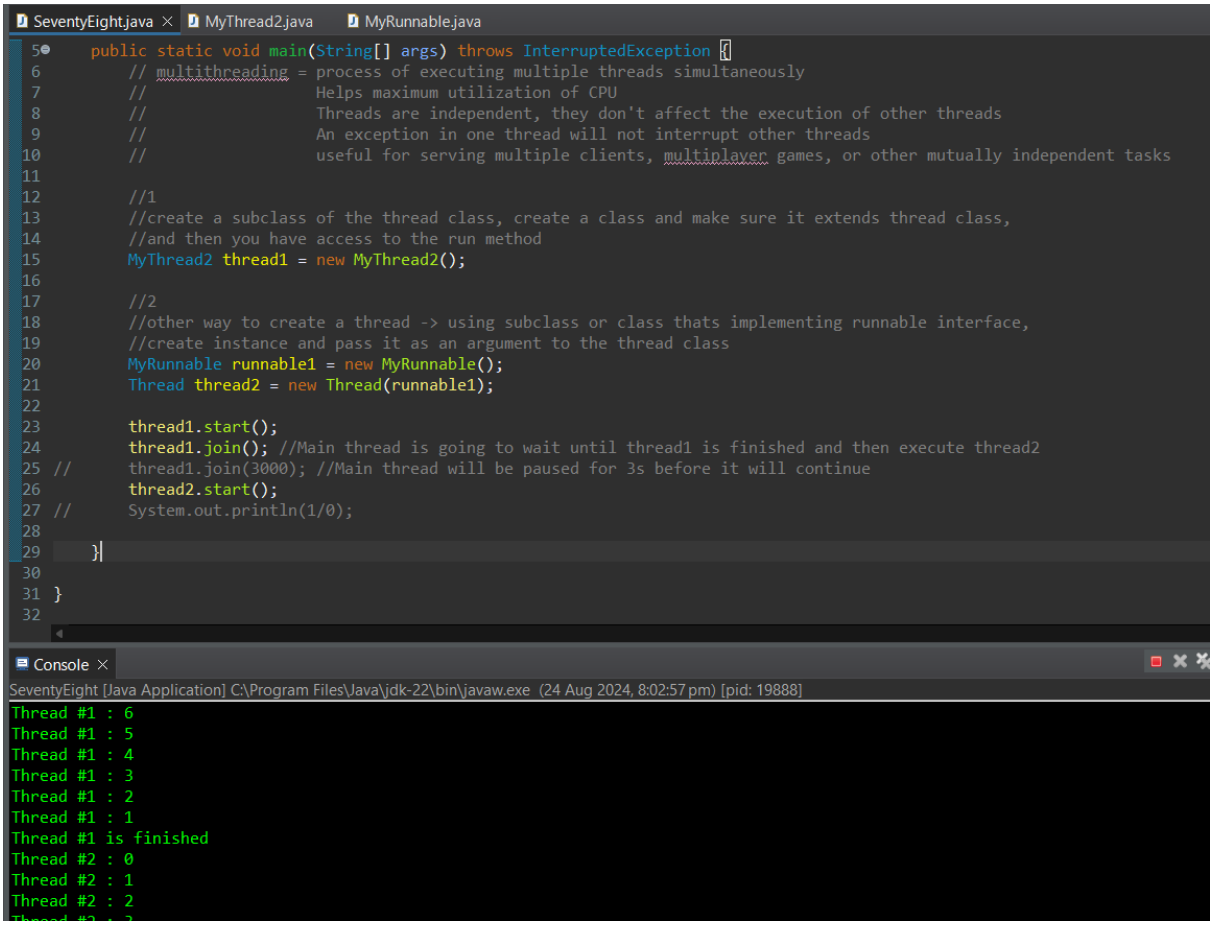
Console

```
<terminated> SeventyEight [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (24 Aug 2024
Thread #1 : 10
Thread #2 : 0
Thread #2 : 1
Exception in thread "Thread-0" java.lang.ArithmeticException: / by zero
        at brocodejava.MyThread2.run(MyThread2.java:15)
Thread #2 : 2
Thread #2 : 3
Thread #2 : 4
Thread #2 : 5
Thread #2 : 6
Thread #2 : 7
Thread #2 : 8
Thread #2 : 9
Thread #2 is finished
```

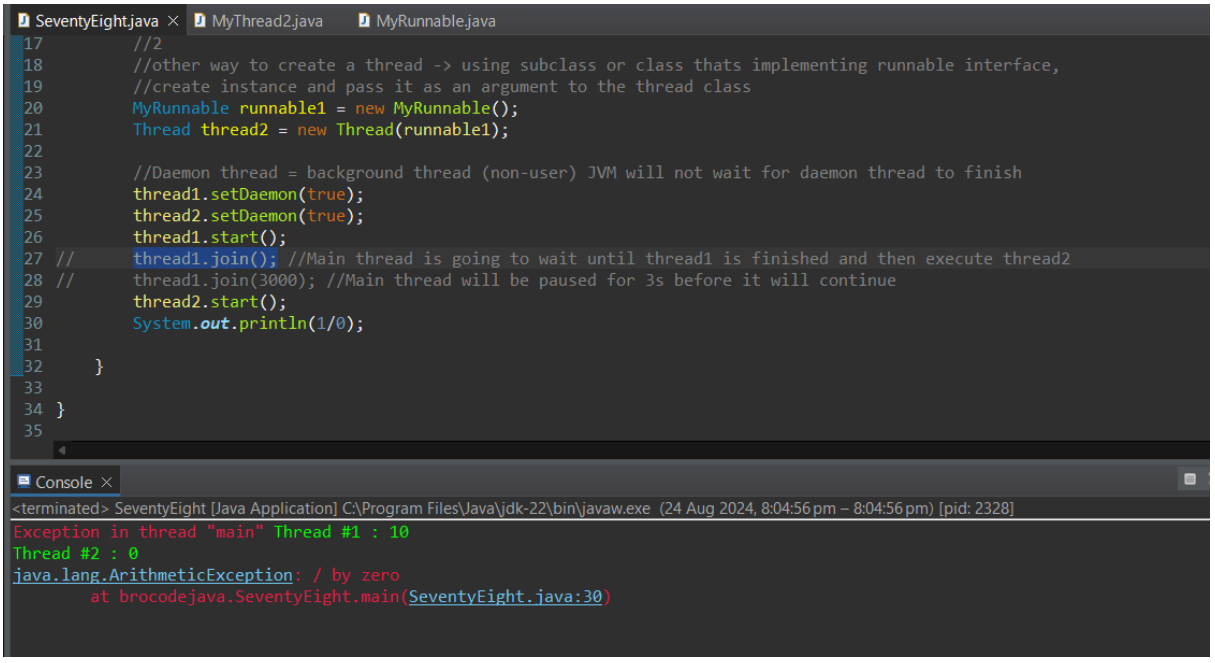

SeventyEight.java
MyThread2.java
MyRunnable.java
//1
//create a subclass of the thread class, create a class and make sure it extends thread class
//and then you have access to the run method
MyThread2 thread1 = new MyThread2();
//2
//other way to create a thread -> using subclass or class thats implementing runnable interfa
//create instance and pass it as an argument to the thread class
MyRunnable runnable1 = new MyRunnable();
Thread thread2 = new Thread(runnable1);
thread1.start();
thread2.start();
System.out.println(1/0);
}
}
Console
ventyEight [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (24 Aug 2024, 7:59:18 pm) [pid: 13768]
read #1 : 10
xception in thread "main" Thread #2 : 0
ava.lang.ArithmeticException: / by zero
at brocodejava.SeventyEight.main(SeventyEight.java:25)
read #1 : 9
read #2 : 1
read #1 : 8
read #2 : 2
read #2 : 3
read #1 : 7

Using join()

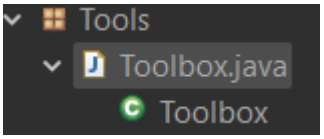

SeventyEight.java
MyThread2.java
MyRunnable.java
1 package brocodejava;
2
3 public class SeventyEight {
4
5 public static void main(String[] args) throws InterruptedException {
6 // multithreading = process of executing multiple threads simultaneously
7 // Helps maximum utilization of CPU
8 // Threads are independent, they don't affect the execution of other threads
9 // An exception in one thread will not interrupt other threads
10 // useful for serving multiple clients, multiplayer games, or other mutually independent tasks
11
12 //1
13 //create a subclass of the thread class, create a class and make sure it extends thread class,
14 //and then you have access to the run method
15 MyThread2 thread1 = new MyThread2();
16
17 //2
18 //other way to create a thread -> using subclass or class thats implementing runnable interface,
19 //create instance and pass it as an argument to the thread class
20 MyRunnable runnable1 = new MyRunnable();
21 Thread thread2 = new Thread(runnable1);
22
23 thread1.start();
24 // thread1.join(); //Main thread is going to wait until thread1 is finished and then execute thread2
25 thread1.join(3000); //Main thread will be paused for 3s before it will continue
26 thread2.start();
27 // System.out.println(1/0);
28
Console
<terminated> SeventyEight [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (24 Aug 2024, 8:02:06 pm – 8:02:19 pm) [pid: 19156]
Thread #1 : 10
Thread #1 : 9
Thread #1 : 8
Thread #2 : 0
Thread #1 : 7
Thread #2 : 1
Thread #1 : 6
Thread #2 : 2
Thread #1 : 5
Thread #2 : 3
Thread #1 : 4
Thread #2 : 4



Daemon thread



79. Packages



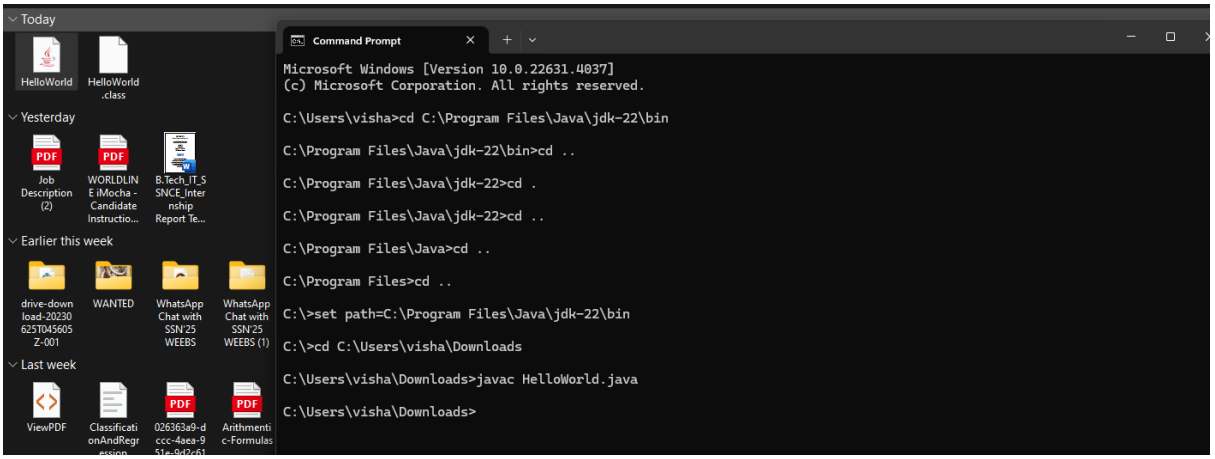

```java
import Tools.Toolbox;
import javax.swing.Icon;
public class Main {
    public static void main(String[] args) {
        Toolbox toolbox = new Toolbox();

    }
}

```

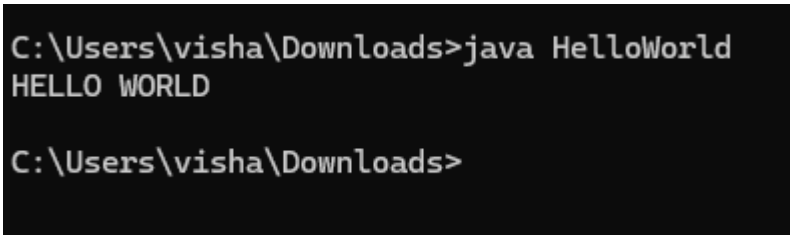
```java
package Tools;

public class Toolbox {

}

```

package name must be mentioned in any class except main

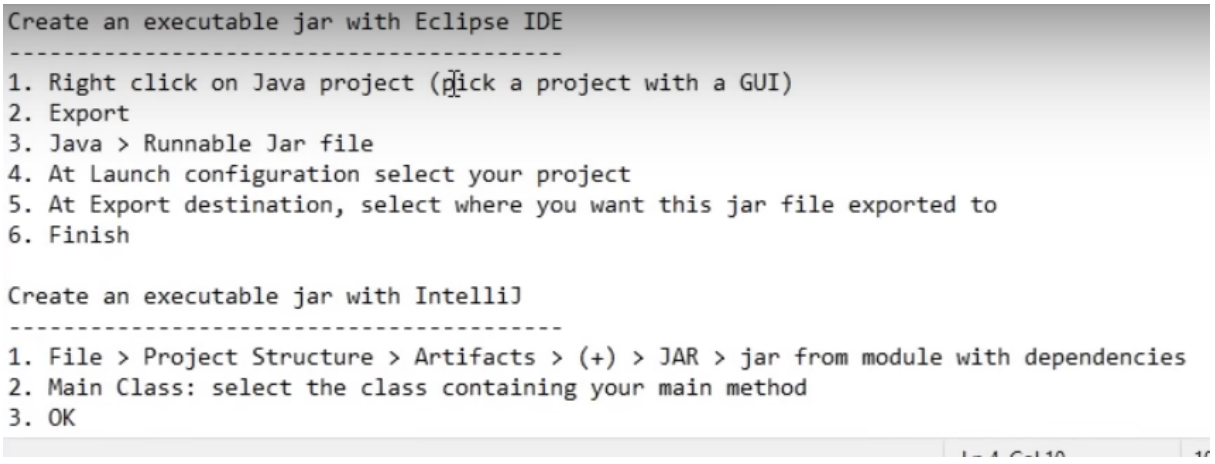
30. Command prompt

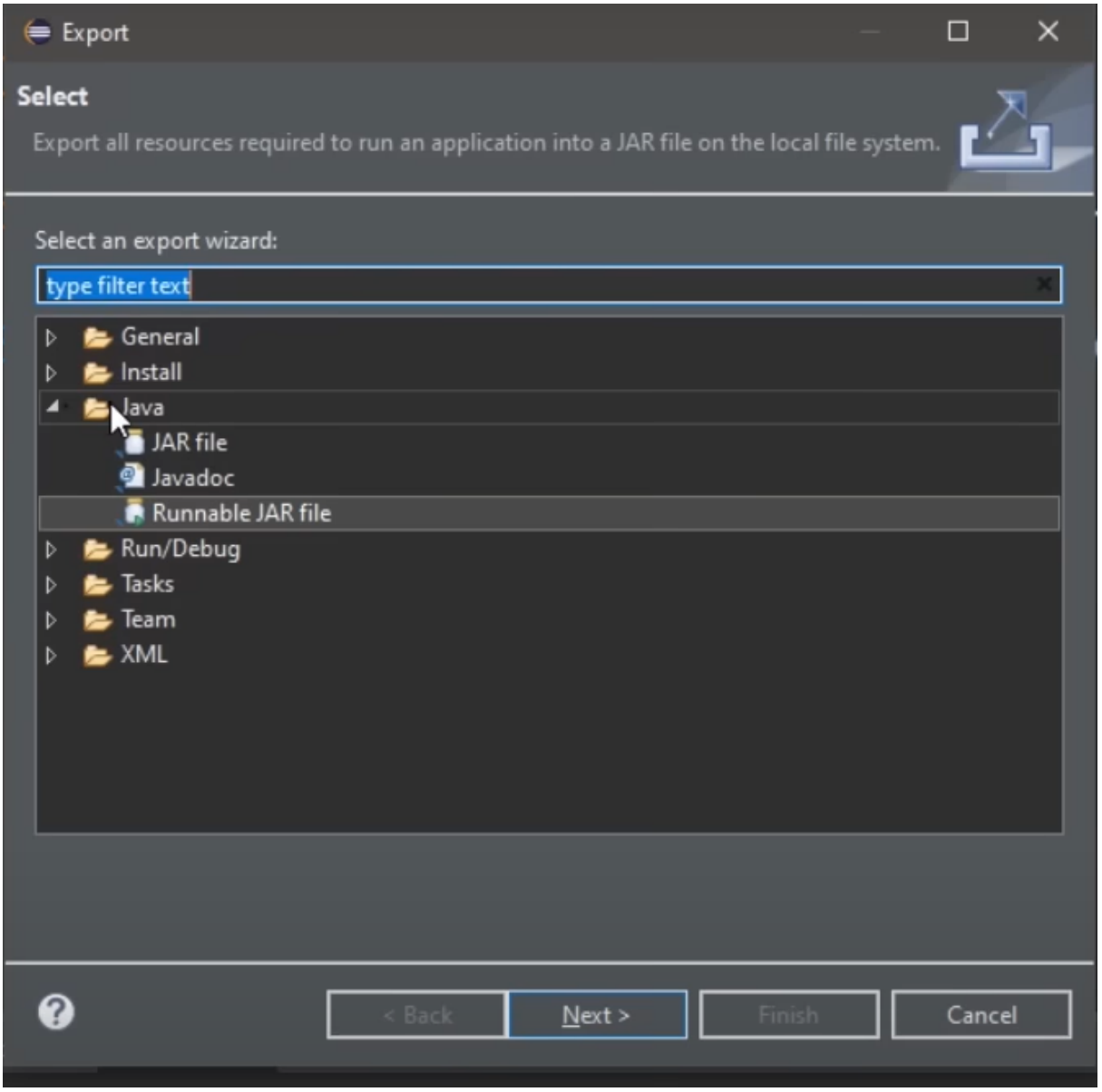
```
compile and run Java with Command Prompt
-----------------------------------------
1. Make sure you have a Java JDK installed (we did this in lesson #1)
2. (optional) use a text editor and save a file as .java
3. Open Command Prompt (windows) or Terminal (Mac) on your computer
4. set path=C:\Program Files\Java\jdk-13.0.1\bin (where JDK is located)
5. cd C:\Users\Cakow\Desktop (or wherever you java file is)
6. javac HelloWorld.java (to compile)
7. java HelloWorld (to run a .class file, it's portable)
```

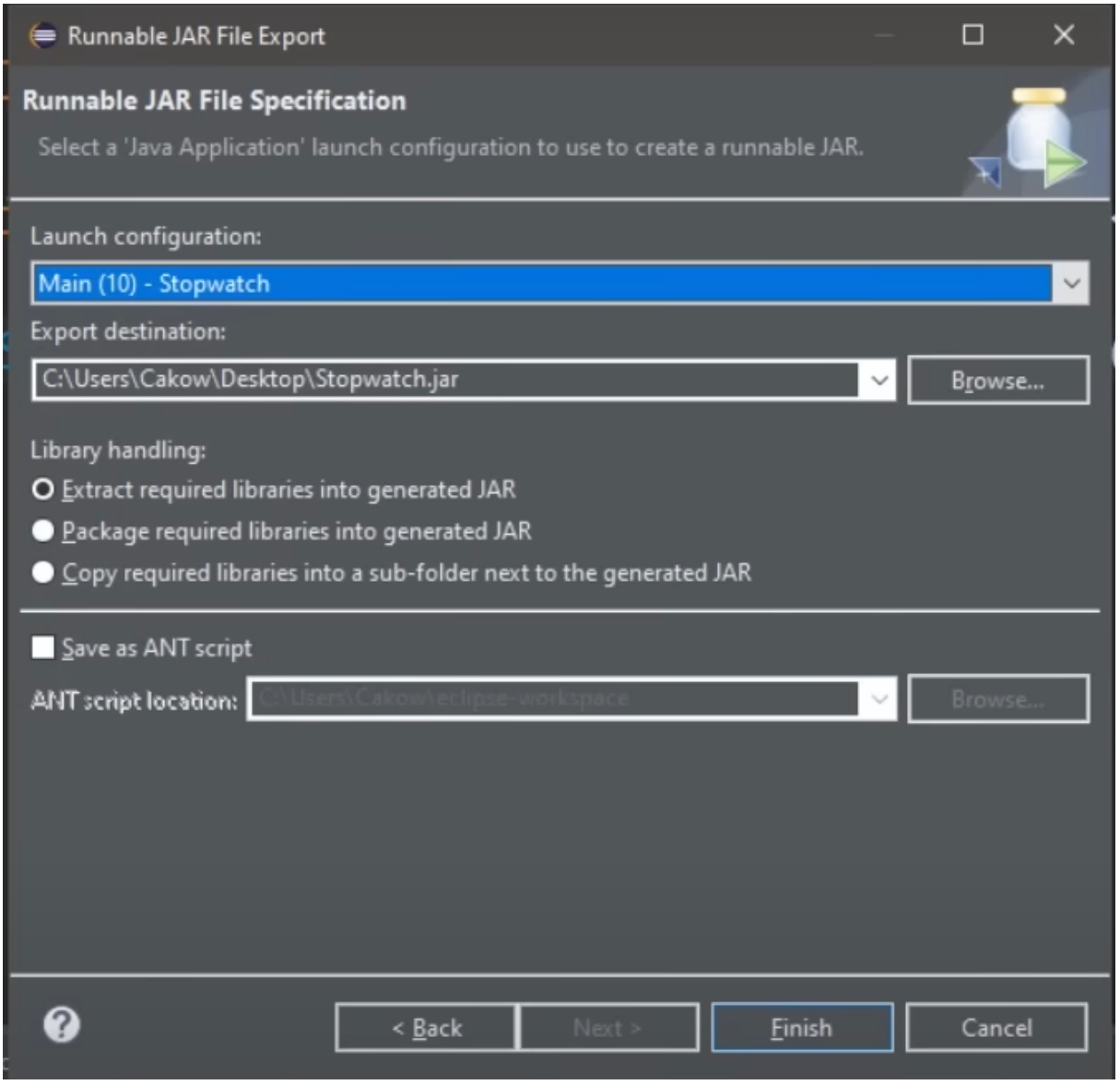




31. executable (.jar file)




Export
Select
Export all resources required to run an application into a JAR file on the local file system.
Select an export wizard:
type filter text
General
Install
Java
JAR file
Javadoc
Runnable JAR file
Run/Debug
Tasks
Team
XML
< Back
Next >
Finish
Cancel


Runnable JAR File Export
Runnable JAR File Specification
Select a 'Java Application' launch configuration to use to create a runnable JAR.
Launch configuration:
Main (10) - Stopwatch
Export destination:
C:\Users\Cakow\Desktop\Stopwatch.jar
Browse...
Library handling:
Extract required libraries into generated JAR
Package required libraries into generated JAR
Copy required libraries into a sub-folder next to the generated JAR
Save as ANT script
ANT script location:
C:\Users\Cakow\eclipse-workspace
Browse...
< Back
Next >
Finish
Cancel