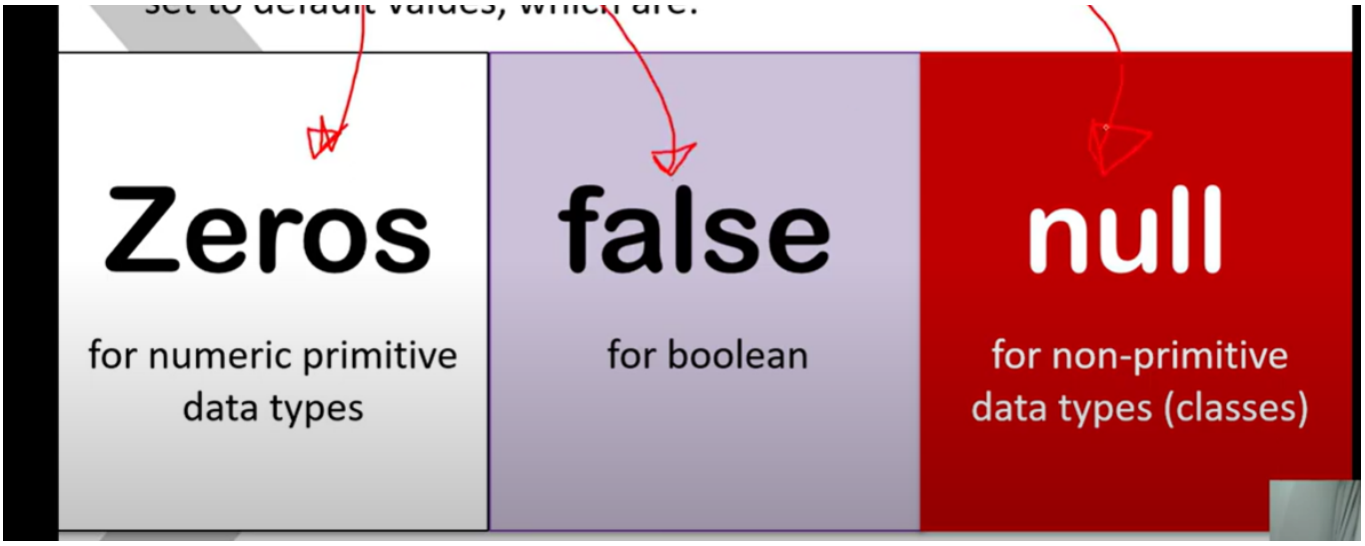
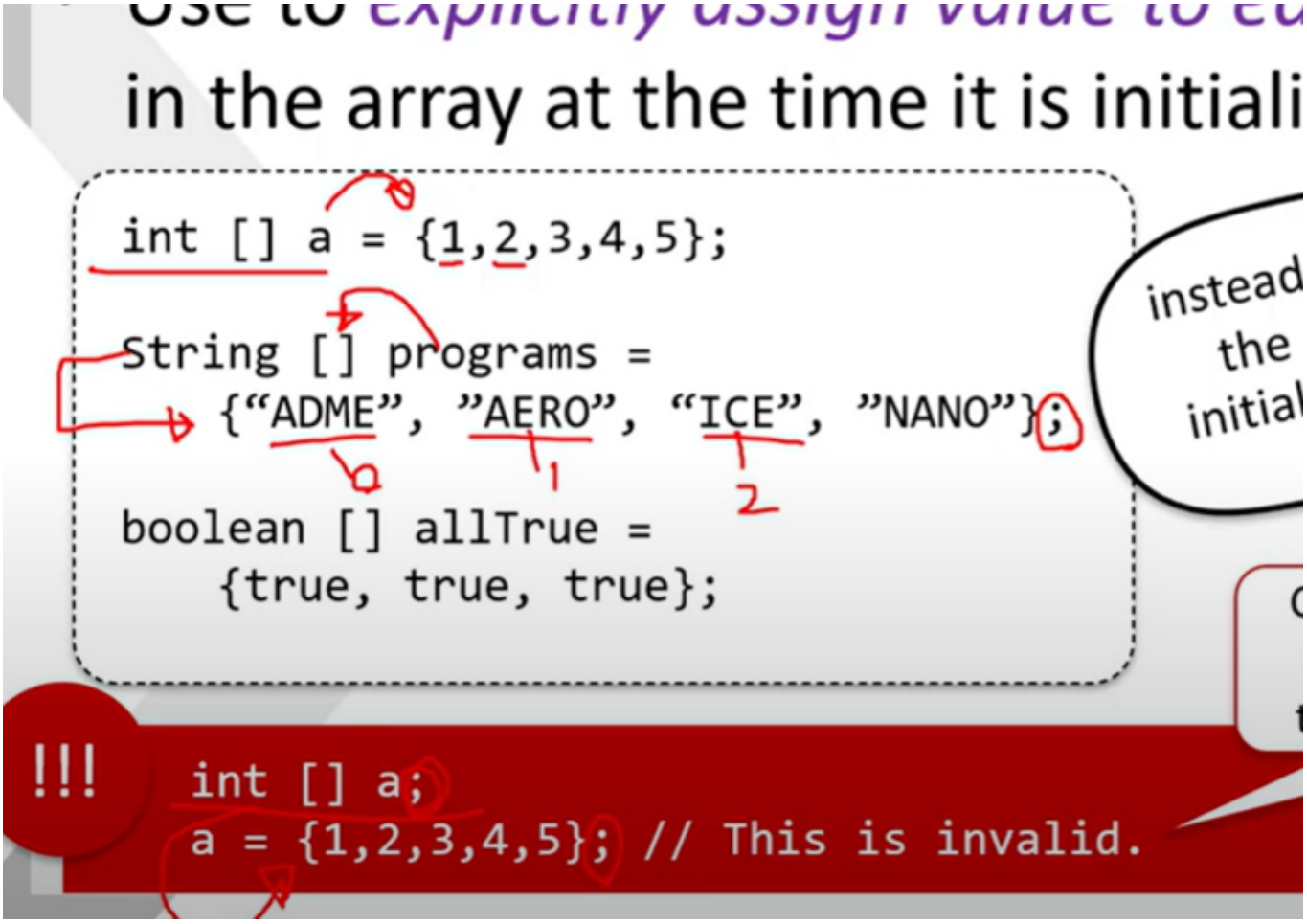
Note

```
Double.phasedouble(String s); // return string as double
```

/ is depends on value

```
char c = String.charAt(index i);
```


Zeros
for numeric primitive data types
false
for boolean
null
for non-primitive data types (classes)


in the array at the time it is initiali
int [] a = {1,2,3,4,5};
String [] programs =
{"ADME", "AERO", "ICE", "NANO"};
boolean [] allTrue =
{true, true, true};
instead
the
initial
!!!
int [] a;
a = {1,2,3,4,5}; // This is invalid.

```
public static void main(String[] args) {
    int [][][] array3D = {
        { {1, 3, 2}, {5, 4}, {7, 6, 8, 9} },
        { {3, 2, 1}, {4, 3, 1, 5} } };

    for (int[][] array2D : array3D) {
        for (int[] array1D : array2D) {
            for (int data : array1D) {
                System.out.print(data + " ");
            }
            System.out.print(", ");
        }
        System.out.println();
    }
}
```

```
input.next(); // Read a String
input.nextInt(); // Read an integer
input.nextDouble(); // Read a double

.nextByte(), .nextShort(), .nextLong(), .nextFloat(), .nextBoolean()
```

input.nextline is read full line even has " "

```java
class Main {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);  // Create a Scanner object
    System.out.println("Enter username");

    String userName = myObj.nextLine();  // Read user input
    System.out.println("Username is: " + userName);  // Output user input
  }
}
```

```java
import java.util.Scanner;

class Main {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);

    System.out.println("Enter name, age and salary:");

    // String input
    String name = myObj.nextLine();

    // Numerical input
    int age = myObj.nextInt();
    double salary = myObj.nextDouble();

    // Output input by user
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
    System.out.println("Salary: " + salary);
  }
}
```

Access modifiers

Specifier	Class	Package	Subclass	World	UML Symbol
private	✓				-
package (default)	✓	✓			~
protected	✓	✓	✓		#
public	✓	✓	✓	✓	+

```
    }

    public boolean equals(Object o) {
        Dice otherDice = (Dice) o;
        if (this.getFaceValue() == otherDice.getFaceValue())
            return true;
        else
            return false;
    }
}
```

Totally different from all SimpleDice Classes.

This equal cannot be used to compare 2 SimpleDice4 objects!!
But can compare SimpleDice4 with any class that can be casted to Dice.

Primitive Data Types

The Java programming language is statically-typed, w

```
int gear = 1;
```

Doing so tells your program that a field named "gear"
other *primitive data types*. A primitive type is predefine

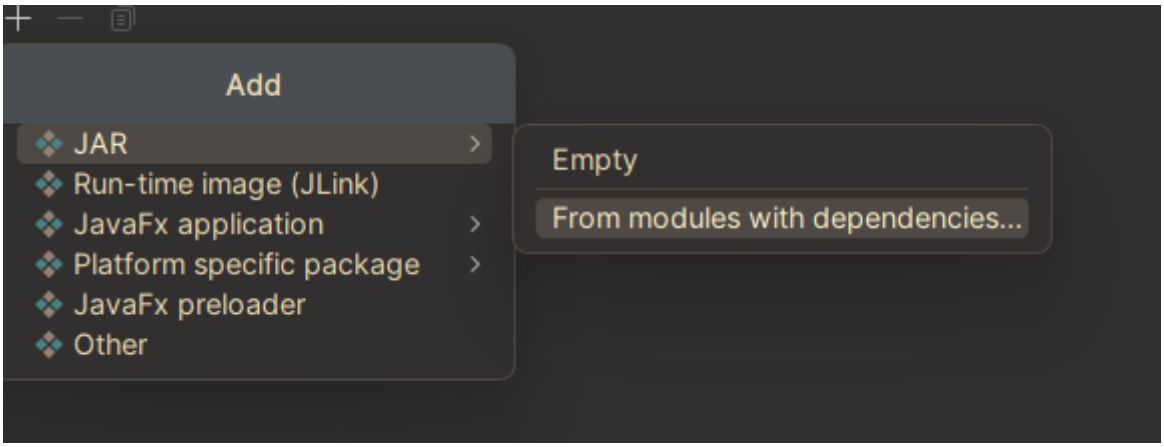
- **byte**: The `byte` data type is an 8-bit signed tw
  place of `int` where their limits help to clarify yo
- **short**: The `short` data type is a 16-bit signed
  savings actually matters.
- **int**: By default, the `int` data type is a 32-bit sig
  maximum value of 2^{32}-1. Use the Integer class
  operations for unsigned integers.
- **long**: The `long` data type is a 64-bit two's com
  value of 2^{64}-1. Use this data type when you ne
- **float**: The `float` data type is a single-precisio
  and `short`, use a `float` (instead of `double`)
  covers `BigDecimal` and other useful classes
- **double**: The `double` data type is a double-pre
  is generally the default choice. As mentioned a
- **boolean**: The `boolean` data type has only two
- **char**: The `char` data type is a single 16-bit Un

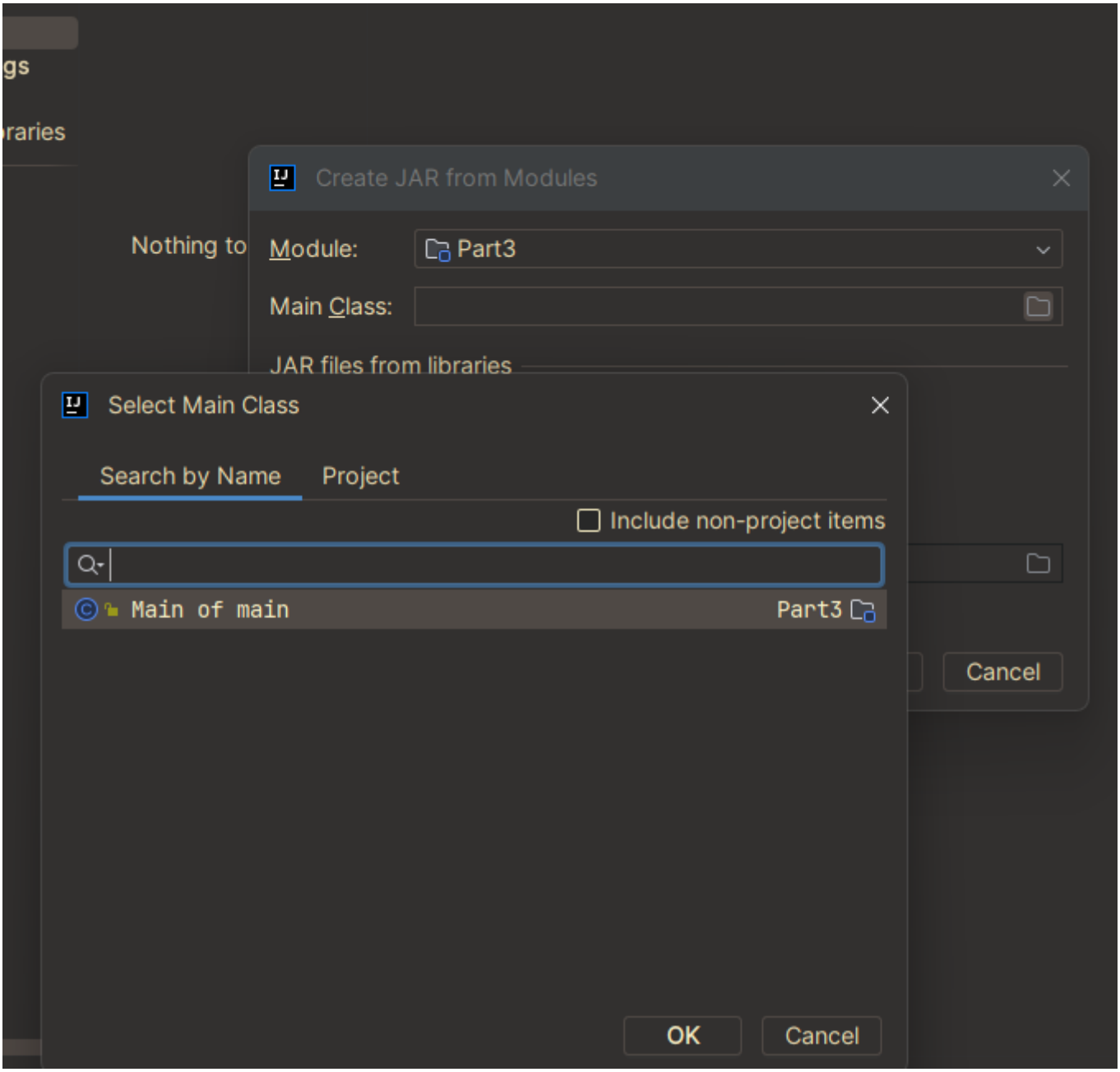
› Primitive type data is passed to a method “by value” (copy), while non-primitive type data is passed to a method “by reference”.
  – Pass by value: there is no change in the passing variable.

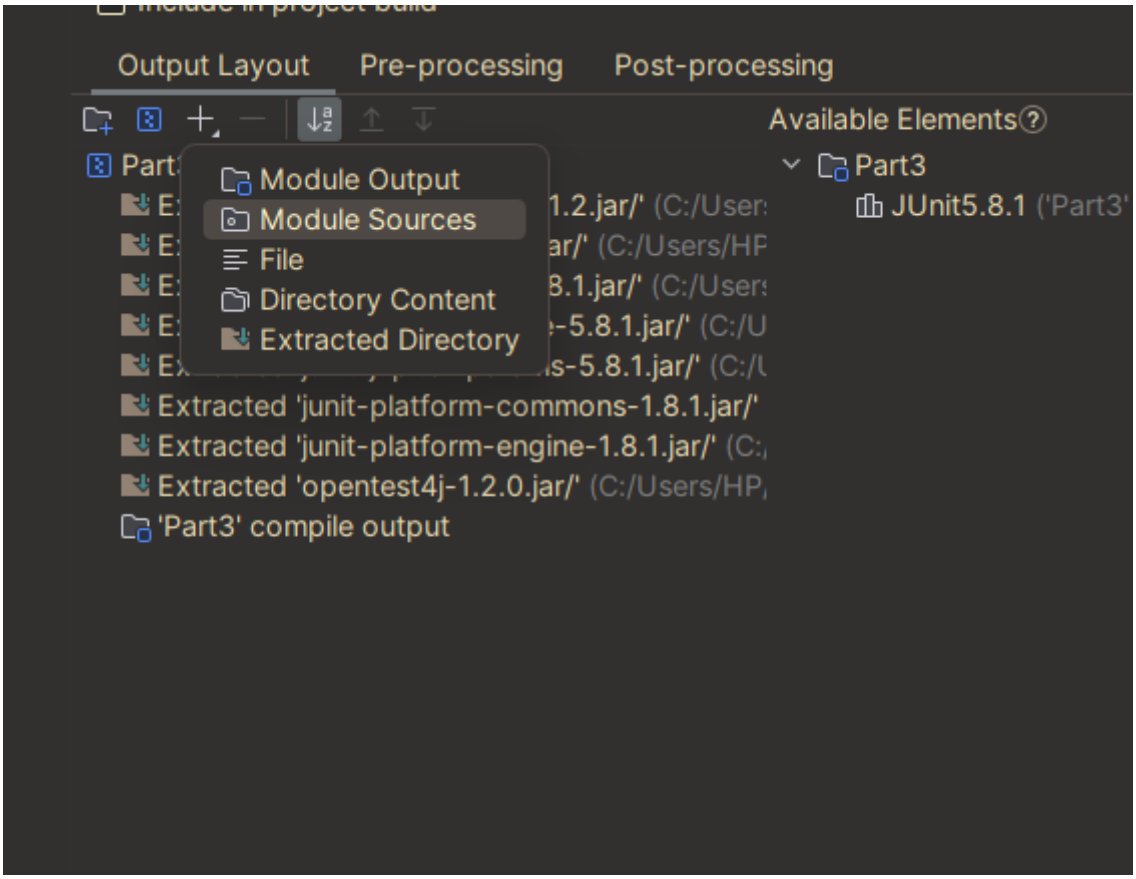
```java
package Exception_Code;

public class MyEX extends Exception {
    String x = "";
    public MyEX(String s) {
        x = s;
        System.out.println("MyException = " + x);
    }
}
```

```java
public class mm {
    public static void main(String[] args) {
        Sup s = new Ch( s: "fff", a: 1);
        System.out.println(s.getClass().getSimpleName()); //ch
        System.out.println(s.p()); //have in super and child -> return child
        System.out.println(s.ch()); // have in only child

    }
}
```

Export jar file


Add
JAR
Run-time image (JLink)
JavaFx application
Platform specific package
JavaFx preloader
Other
Empty
From modules with dependencies...


Create JAR from Modules
Module:
Part3
Main Class:
JAR files from libraries
Nothing to
Cancel
Select Main Class
Search by Name
Project
Include non-project items
Main of main
Part3
OK
Cancel




```
3> java -jar .\Part3.jar
```

o

parse

```java
int number;

try {
    number = Integer.parseInt(validString);
    System.out.println("Converted integer: " + number);

    number = Integer.parseInt(invalidString);
    System.out.println("Converted integer: " + number);
} catch (NumberFormatException e) {
    System.out.println("Invalid integer input");
}
```

Scanner

```java
public static void main(String[] args) {
    File fileToRead = new File("duplist.txt");


    //TODO: FILL CODE
    Scanner sc = null;
    try {
        sc = new Scanner(fileToRead);
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
    while(sc.hasNextLine()){
        String[] sp = sc.nextLine().split(" ");
        System.out.println(sp[0]);
        System.out.println(Integer.parseInt(sp[1]));
    }
```

```java
eTurns(int currentChargeTurns) {
= Math.max(0,Math.min(getMaxChargeTurns(), currentChargeTurns));
```

- abstract can't be constructor

** Noted that Access Modifier Notations can be listed below*

+ (public)
# (protected)
- (private)
Underline (static)
Italic (abstract)

Set-Up Instruction