**P 1 How to Print an Integer entered by an user**

import java.util.Scanner;

public class HelloWorld {

public static void main(String[] args) {

// Creates a reader instance which takes

// input from standard input - keyboard

Scanner reader = new Scanner(System.in);

System.out.print("Enter a number: ");

// nextInt() reads the next integer from the keyboard

int number = reader.nextInt();

// println() prints the following line to the output screen

System.out.println("You entered: " + number);

}

}

**Output**

Enter a number: 10

You entered: 10

In this program, an object of Scanner class, reader  is created to take inputs from standard input, which is keyboard.

Then, Enter a number prompt is printed to give the user a visual cue as to what they should do next.

reader.nextInt()then reads all entered integers from the keyboard unless it encounters a new line character \n (Enter). The entered integers are then saved to the integer variable number.

If you enter any character which is not an integer, the compiler will throw an InputMismatchException.

Finally, number is printed onto the standard output (System.out) - computer screen using the function println().

**Difference between println(), print() and printf()**

* print() - It prints string inside the quotes.
* println() - It prints string inside the quotes similar like print() method. Then the cursor moves to the beginning of the next line.
* printf() - It provides string formatting (similar to [printf in C/C++ programming](https://www.programiz.com/cpp-programming/library-function/cstdio/printf" \o "printf in C/C++ programming)).

**P2 Example: print() and println()**

class Output {

public static void main(String[] args) {

System.out.println("1. println ");

System.out.println("2. println ");

System.out.print("1. print ");

System.out.print("2. print");

}

}

**Output**:

1. println

2. println

1. print 2. print

In the above example, we have shown the working of the print() and println() methods. To learn about the printf() method, visit [Java printf()](https://www.cs.colostate.edu/~cs160/.Summer16/resources/Java_printf_method_quick_reference.pdf).

**P 3 Example: Printing Variables and Literals**

class Variables {

public static void main(String[] args) {

Double number = -10.6;

System.out.println(5);

System.out.println(number);

}

}

When you run the program, the output will be:

5

-10.6

Here, you can see that we have not used the quotation marks. It is because to display integers, variables and so on, we don't use quotation marks.

### P4 Get Integer Input From the User

import java.util.Scanner;

class Input {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter an integer: ");

int number = input.nextInt();

System.out.println("You entered " + number);

// closing the scanner object

input.close();

}

}

**Output**:

Enter an integer: 23

You entered 23

In the above example, we have created an object named input of the Scanner class. We then call the nextInt() method of the Scanner class to get an integer input from the user.

Similarly, we can use nextLong(), nextFloat(), nextDouble(), and next() methods to get long, float, double, and string input respectively from the user.

### P5 Get float, double and String Input

import java.util.Scanner;

class Input {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Getting float input

System.out.print("Enter float: ");

float myFloat = input.nextFloat();

System.out.println("Float entered = " + myFloat);

// Getting double input

System.out.print("Enter double: ");

double myDouble = input.nextDouble();

System.out.println("Double entered = " + myDouble);

// Getting String input

System.out.print("Enter text: ");

String myString = input.next();

System.out.println("Text entered = " + myString);

}

}

**Output**:

Enter float: 2.343

Float entered = 2.343

Enter double: -23.4

Double entered = -23.4

Enter text: Hey!

Text entered = Hey!

## P 6 Find ASCII value of a character

public class AsciiValue {

public static void main(String[] args) {

char ch = 'a';

int ascii = ch;

// You can also cast char to int

int castAscii = (int) ch;

System.out.println("The ASCII value of " + ch + " is: " + ascii);

System.out.println("The ASCII value of " + ch + " is: " + castAscii);

}

}

**Output**

The ASCII value of a is: 97

The ASCII value of a is: 97

In the above program, character a is stored in a char variable, ch. Like, double quotes (" ") are used to declare strings, we use single quotes (' ') to declare characters.

## P6 Print an Array using For loop

public class Array {

public static void main(String[] args) {

int[] array = {1, 2, 3, 4, 5};

for (int element: array) {

System.out.println(element);

}

}

}

**Output**

1

2

3

4

5

## P7 Concatenate Two Arrays using arraycopy

import java.util.Arrays;

public class Concat {

public static void main(String[] args) {

int[] array1 = {1, 2, 3};

int[] array2 = {4, 5, 6};

int aLen = array1.length;

int bLen = array2.length;

int[] result = new int[aLen + bLen];

System.arraycopy(array1, 0, result, 0, aLen);

System.arraycopy(array2, 0, result, aLen, bLen);

System.out.println(Arrays.toString(result));

}

}

**Output**

[1, 2, 3, 4, 5, 6]

# P8 Program to Check if a String is Empty or Null

class Main {

public static void main(String[] args) {

// create null, empty, and regular strings

String str1 = null;

String str2 = "";

String str3 = " ";

// check if str1 is null or empty

System.out.println("str1 is " + isNullEmpty(str1));

// check if str2 is null or empty

System.out.println("str2 is " + isNullEmpty(str2));

// check if str3 is null or empty

System.out.println("str3 is " + isNullEmpty(str3));

}

// method check if string is null or empty

public static String isNullEmpty(String str) {

// check if string is null

if (str == null) {

return "NULL";

}

// check if string is empty

else if(str.isEmpty()){

return "EMPTY";

}

else {

return "neither NULL nor EMPTY";

}

}

}

**Output**

str1 is NULL

str2 is EMPTY

str3 is neither NULL nor EMPTY

**P9 WAP in java for Method Overloading**

1. **class** Adder{
2. **static** **int** add(**int** a,**int** b){**return** a+b;}
3. **static** **int** add(**int** a,**int** b,**int** c){**return** a+b+c;}
4. }
5. **class** TestOverloading1{
6. **public** **static** **void** main(String[] args){
7. System.out.println(Adder.add(11,11));
8. System.out.println(Adder.add(11,11,11));
9. }}

Output:

22

33

**P10 WAP in java for Method Overridding**

1. **class** Vehicle{
2. //defining a method
3. **void** run(){System.out.println("Vehicle is running");}
4. }
5. //Creating a child class
6. **class** Bike2 **extends** Vehicle{
7. //defining the same method as in the parent class
8. **void** run(){System.out.println("Bike is running safely");}
10. **public** **static** **void** main(String args[]){
11. Bike2 obj = **new** Bike2();//creating object
12. obj.run();//calling method
13. }
14. }

Output:

Bike is running safely

**P11WAP in java for single inheritance**

1. **class** Animal{
2. **void** eat(){System.out.println("eating...");}
3. }
4. **class** Dog **extends** Animal{
5. **void** bark(){System.out.println("barking...");}
6. }
7. **class** TestInheritance{
8. **public** **static** **void** main(String args[]){
9. Dog d=**new** Dog();
10. d.bark();
11. d.eat();
12. }}

Output:

barking...

eating...

**P12 WAP in java for multilevel inheritance**

1. **class** Animal{
2. **void** eat(){System.out.println("eating...");}
3. }
4. **class** Dog **extends** Animal{
5. **void** bark(){System.out.println("barking...");}
6. }
7. **class** BabyDog **extends** Dog{
8. **void** weep(){System.out.println("weeping...");}
9. }
10. **class** TestInheritance2{
11. **public** **static** **void** main(String args[]){
12. BabyDog d=**new** BabyDog();
13. d.weep();
14. d.bark();
15. d.eat();
16. }}

Output:

weeping...

barking...

eating...

**P13 WAP in java for multiple inheritance**

interface AnimalEat {

   void eat();

}

interface AnimalTravel {

   void travel();

}

class Animal implements AnimalEat, AnimalTravel {

   public void eat() {

      System.out.println("Animal is eating");

   }

   public void travel() {

      System.out.println("Animal is travelling");

   }

}

public class Demo {

   public static void main(String args[]) {

      Animal a = new Animal();

      a.eat();

      a.travel();

   }

}

## Output

Animal is eating

Animal is travelling

P14 WAP in java for exception handling

1. **public** **class** JavaExceptionExample{
2. **public** **static** **void** main(String args[]){
3. **try**{
4. //code that may raise exception
5. **int** data=100/0;
6. }**catch**(ArithmeticException e){System.out.println(e);}
7. //rest code of the program
8. System.out.println("rest of the code...");
9. }
10. }

**Output:**

Exception in thread main java.lang.ArithmeticException:/ by zero

rest of the code...

P15 WAP in java for exception handling for Finally Block

1. **lass** TestFinallyBlock {
2. **public** **static** **void** main(String args[]){
3. **try**{
4. //below code do not throw any exception
5. **int** data=25/5;
6. System.out.println(data);
7. }
8. //catch won't be executed
9. **catch**(NullPointerException e){
10. System.out.println(e);
11. }
12. //executed regardless of exception occurred or not
13. **finally** {
14. System.out.println("Finally block is always executed");
15. }
17. System.out.println("Rest of the code...");
18. }
19. }

**Output:**

**5**

**Finally block is always executed**

**Rest of the code…**