|  |  |
| --- | --- |
| **Compiler** | **Interpreter** |
| 1.Scan the entire program and translate it into machine code | 1.Translate program one statement at a time |
| 2.Compiler will take large amount of time to analyze the code | 2.Interpreter will takes less amount of time to analyze the code. |
| 3.However overall execution time and complexity is faster than interpreter | 3.However overall execution time and complexity is slower than compiler |
| 4.Generate object code which further queries lining hence required more memory | 4.No object code is generated hence are more efficient. |
| 5.Programing language like c, c++, java uses compiler. | 5.Programing like JavaScript, Ruby, and Python uses interpreter |
| 6.Compiler Follows Source Code to object code to Machine | 6.Interpreter follows Source Code to Intermediate Code to Interpreter |
| 7.It is best suited for the Production Environment | 7.It is best suited for the program and development environment. |
| 8.Compliers generates intermediate machine code | 8.Interpreter never generate any intermediate machine code |
| 9.Display all errors after, compilation, all at the same time | 9.Displays all errors of each line one by one |
|  |  |

**2)** **Few details of strongly typed language and loosely typed language minimum 4 examples?**

**A) Strong Type Language: -**

**1)** strongly typed language checks the type of a variable before performing an operation on it

**2)** A strongly typed language has stricter typing rules at compile time, which implies that errors and exceptions are more likely to happen during compilation

**3)** Most of these rules affect variable assignment, return values and function calling

**4)** For instance, Java is a strongly typed language because whenever you perform an operation on an object, Java checks the type of the object.

**Example:-** JAVA,Python can be both dynamically and strongly typed.

**Loosely Typed Language: -**

1. weakly typed language does not check the type of a variable before performing an operation on it
2. weakly typed languages perform implicit casts

Example: C Language, C++,.

1. **Work on data types? Write a Programs for Datatypes in java?**

**Source Code: -**

**package** demo\_one;

**public** **class** Datatypes {

**public** **static** **void** main(String[] args)

{

**int** num=10;

**double** numOne=20.0;

String str="praveen";

**char** ch='s';

**float** fot=1.1f;

**boolean** boll = **true**;

System.***out***.println(num);

System.***out***.println(numOne);

System.***out***.println(str);

System.***out***.println(ch);

System.***out***.println(fot);

System.***out***.println(boll);

}

}

**O/P:**

10

20.0

praveen

s

1.1

true

1. **Write the simplest code for the If, while ,for, switch?**

**For:**

**package** demo\_one;

**public** **class** ForExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** n = 5;

**for** (**int** i = 1; i <= n; ++i) {

System.***out***.println("Java is fun");

}

}

}

**O/P:**

Java is fun

Java is fun

Java is fun

Java is fun

Java is fun

**While:**

**package** demo\_one;

**public** **class** WhileExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** i = 1, n = 5;

**while**(i <= n) {

System.***out***.println(i);

i++;

}

}

}

**O/P:**

1

2

3

4

5

**IF-Condition:**

**package** demo\_one;

**public** **class** IfelseExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** number = 13;

**if** (number%2==0)

{

System.***out***.println("even number");

}

**else**{

System.***out***.println("odd number");

}

}

}

O/P:

odd number

**SWITCH Program:**

**package** demo\_one;

**public** **class** SwitchCaseExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** day = 5;

String dayString;

// switch statement with int data type

**switch** (day) {

**case** 1:

dayString = "Monday";

**break**;

**case** 2:

dayString = "Tuesday";

**break**;

**case** 3:

dayString = "Wednesday";

**break**;

**case** 4:

dayString = "Thursday";

**break**;

**case** 5:

dayString = "Friday";

**break**;

**case** 6:

dayString = "Saturday";

**break**;

**case** 7:

dayString = "Sunday";

**break**;

**default**:

dayString = "Invalid day";

}

System.***out***.println(dayString);

}

}

**O/P:**

Friday

1. **In a class create constructor and object use data members and function members?**

**Source Code:**

**package** demo\_one;

**public** **class** Student {

String name;

String course;

**int** age;

**public** Student(String name, String course,**int** age)

{

**this**.name = name;

**this**.course = course;

**this**.age = age;

}

**public** String getName()

{

**return** name;

}

**public** **static** **void** main(String[] args)

{

Student s1 = **new** Student("Praveen","CSE",23);

System.***out***.println(s1.getName());

}

}

**O/P:**

Praveen

1. Create a class create 2 constructors in a single class use both parameterizes and non-parametrized?

**Source Code:**

**package** demo\_one;

**public** **class** Mobile {

String model,clr;

**int** price;

Mobile(){

}

Mobile(String c){

clr=c;

}

Mobile(String c, String m, **int** p)

{

clr=c;model=m;price=p;

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Mobile m1 = **new** Mobile();

System.***out***.println(m1.clr+" "+m1.model+" "+m1.price);

Mobile m2 = **new** Mobile("grey");

System.***out***.println(m2.clr+" "+m2.model+" "+m2.price);

Mobile m3 = **new** Mobile("red","iphone",50000);

System.***out***.println(m3.clr+" "+m3.model+" "+m3.price);

}

}

**O/P:** null null 0

grey null 0

red iphone 50000