|  |  |
| --- | --- |
| **Compiler** | **Interpreter** |
| 1.It is best suited for the Production Environment | 1.Interpreter follows Source Code to Intermediate Code to Interpreter |
| 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.Generates object code which further requires linking,hence requires more memory. | 5.No object code is generated,hence are memory efficient. |
| 6.Compiler Follows Source Code to object code to Machine | 6.translate program one statement at a time |
| 7.Scan the entire program and translate it into machine code | 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 |
| 10.Programing language like c, c++, java uses compiler. | 10.Programing like JavaScript, Ruby, and Python uses interpreter |

**2)**

**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.

1. **Loosely Typed Language: -**
2. weakly typed language does not check the type of a variable before performing an operation on it
3. weakly typed languages perform implicit casts

Example: C Language, C++,.

3)

**if:**

**public** **class** exaif

{

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

{

**int** age=24;

**if**(age>18)

{

System.***out***.println("Age is greater than 18");

}

**Else{**

System.***out***.println("age is lesser than 18"); }}}

**While:**

**public** **class** whil

{

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

{

**int** i=1;

**do** {

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

i++;

}

**while**(i<=10);

}

**for:**

**public** **class** fo

{

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

{

**for**(**int** i=0;i<=10;i++)

{

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

}

}

}

**switch case:**

**public** **class** switchcase

{

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

{ **int** number=20;

**switch**(number)

{

**case** 10:System.***out***.println("10");

**break**;

**case** 20:System.***out***.println("20");

**break**;

**case** 30:System.***out***.println("30");

**break**;

**default**:System.***out***.println("not in 10,20,30");

}}}

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

**Source Code:**

package demo;

public class cons

{

public static int one;

public static double points;

public Constructor(int one,double points)

{

this.o*ne*=one;

this.points=points;

}

public void display()

{

System.***out***.println("integer "+o*ne*+" double "+points+");

}

public static void main(String[] args)

{

Constructor c1 = new Constructor(10,10.0);

c1.display();

}

}

O

package prg2;

public class demo2 {

static String *str*;

public demo2()

{

*str*="ASH";

}

public demo2(String str) {

this.*str* = str;

}

public static void main(String[] args)

{

demo2 obj = new demo2("ASH");

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

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

}

}