(3) Write a sample program to get use of Abstraction and interface

Code (sample usage):

```
class Test {
 Sys (" A Normal class in JAVA");
}
// creating Abstract class
abstract class First {
  // Data Members
   int age = 22;
   int name = 'AADI';
// Data Methods
 void learn() {
 Sys (" learning JAVA in NIT");
 }
abstract void ();
}
public class abstract_class {
  public static void main (String args[]) {
```

```
First First = new First();
First.learn();
}
}
```

Interface

```
Interface I1 {
// unimplemented methods
 void walk();
abstract void Run();
}
class sample implements I1 {
  void walk() {
  Sys ("I'm walking");
  void Run() {
  Sys ("Aadi is Running");
  }
}
Public class InterfaceExample {
  psvm (String args[]) {
```

// allows only abstract methods / unimplemented

```
Sample sample = new Sample();
sample.walk();
sample.Run();
}
}
```

(5) write a sample java code for

1) local 2) global 3) static

```
class localvariable {
  void sample() {
    String name = 'NIT';
    int marks = '999';
  }
}
```

} local variables scope inside that method only

```
class instance_variable {
  int fee = 18000;
  String Institute = "NIT";
  void career() {
  Sys ("Total fee" + fee);
  }
}
```

} global variables scope accessable inside the class and thier methods

```
class static_variable {
    static int age = 22;
    static float grade = 92.3;
    void bio() {
        sys("Im " +age+ "older");
    }
// must use "static" keyword to get use of
static variable
public class variables {
    psvm (String args[]) {
        localvariable lv = new localvariable();
        lv.samples();
        instance_variable iv = new instance_variable();
        iv.career();
        static_variable sv = new static_variable();
        sv.bio();
    }
}
```

static variables scope like instance variables

① multiple inheritance NOT ALLOWED and Alternate solution

```
class Father {
  // code
}
class mother {
  // code
}
class child child
class child extends Father, mother {
}
```

✗ won't work due to Ambiguity problem

// alternate for that interface

// Interface also act as a class

```
interface I1 {          // like father class
  void property();
  void Assets();
}
interface I2 {
  void Building();   // like mothers class
}
class child implements I1, I2 {
  void property() {
  sys ("I have 2cr property");
  }
  void Assets() {
  sys("I have passive income");
   }
  void Building() {
   sys("I have an Apartment in Hyd");
  }
```

```
public class MI {

    psvm (String args[]) {

        Child child = new Child();

        child.propesty();   } from parent 1
        child.Assets();

        child.Building(); } from parent 2

}

}
```

(2) Star pattern

```
        *
      * * *
    * * * * *
```

```
class Star pattern {

    for (int i=5; i>=0; i--) {

    for (int j=0; j<=i(2)*; j++) {

        Sys(" ");

    }

     Sys ("*");

     Sysln ( );   }
```

```
public class Stat {

psvm (String args[]) {

Stat patter sp= new Stat pattern();

}
}
```