Assesment - I;

1. write a program to show why java doesn't support multiple inheritance & how to resolve that issue (with seperate programs)

Sol:

```
Public class MultipleInheritance
{

}
class A
{ int a=10;
  void aa()
  { sop("Hi I am object of A");
  }
}
class B
{ int b=11;
  void bb()
  { sop("Hi I am object of B");
  }
}
class C extends A,B
{

}
```

(Ambiguity Error)

Can resolve by interface.

```
Public class MultipleInheritanceByInterface
{ public static void main(String[] args)
  { C c1 = new C();

    c1.aa();
    c1.bb();
    c1.cc();
  }
}
```

```
interface A
{
   void aa();
}
interface B
{
   void bb();
}
```

```
class C implements A,B
{
public void aa()
   {
     sop("method of A");
   }
public void bb()
   {
     sop("method of B");
   }
   void cc()
   {
     sop("method of c");
   }
}
```

---

(2) print

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

```
class EquilateralTriangle
{
  public static void main(String[] args)
  {
    int n = 3;
    for(int i=1; i<=n; i++)
    {
      for(int j=1; j<=n-i; j++)
      {
        sop(" ");//spacebar
      }
      for(int k=1; k<=2i-1; k++)
      {
        sop("*");
      }
    }
  }
}
```

③ Abstraction & Interface:-

Abstraction

```
Public class Abstraction
{ public static void main(String[] args)
  {
      B b = new B();
      b.aa();
      b.bb();
  }
}
abstract class A
{
  void aa()
  { sop("I am aa method");
  }
  abstract void bb();
}
class B extends A
{
  public void bb()
  {
    sop("I am bb method");
  }
}
```

Interfaces

```
Public class Interface
{
public static void main(String[] args)
{ B b = new B();
  b.aa();
}
}
interface A
{ void aa();
}
class B implements A
{
  public void aa{
    sop("I am aa");
  }
}
```

④ Parametrized constructor.

```
Public class Login
{ String name;
  Login()
  {
    sop("Hi Guest");
  }
  Login(String n)
  {
    name=n;
    sop("Hi"+name);
  }
Public static Void main(String[] args)
{
  Login l1 = new Login();
  Login l2 = new Login("Lokesh");
}
}
```

If we give parameter then object can be distinctly identified.

(5) Static variable, instance variable, local variable

```
Public class Variables
{ int a = 10;
  static int b = 10;

  void add()
  { int d = a+b;
    sop(d); // 20
  }

  void increment()
  { a=a+1;
    b=b+1; }

  static void mul()
  {
    // int c = a*b; (X) Since a is instance variable.
    // sop(d); (X) since d is local variable of add method.
    sop("mul");
  }

  public static void main(String[] args)
  {
    Variables v1 = new Variables();
    sop(v1.a);          // 10
    v1.add();           // 20
    v1.increment();
    Variables.mul();    // mul
    sop(v1.a);          // 11
    sop(Variables.b);   // 11

    Variables v2 = new Variables();
    sop(v2.a);          // 10   // 20
    v2.add();
    v2.increment();
    Variables.mul();    // mul
    sop(v2.a);          // 11
    sop(Variables.b);   // 12
  }
}
```

Static variable will be in the memory when running.