

## Lesson-13: Inheritance & Polymorphism

### Inheritance & Polymorphism

#### Program 1:

```
class Teacher:

    def teach(self):
        print("I randomly teach any subject")

    def schoolname(self):
        print("I work in Subhash Programming Classes")

class CTeacher(Teacher):
    pass

class CppTeacher(Teacher):
    pass

c = CTeacher()
c.teach()
c.schoolname()

cpp = CppTeacher()
cpp.teach()
cpp.schoolname()
```

#### **Output:**

```
I randomly teach any subject
I work in Subhash Programming Classes
I randomly teach any subject
I work in Subhash Programming Classes
```

#### Program 2:

```
class Teacher:

    def teach(self):
        print("I randomly teach any subject")

    def schoolname(self):
        print("I work in Subhash Programming Classes")

class CTeacher(Teacher):
    def teach(self):
        print("I teach C Programming")

class CppTeacher(Teacher):
    def teach(self):
        print("I teach C++ Programming")
```

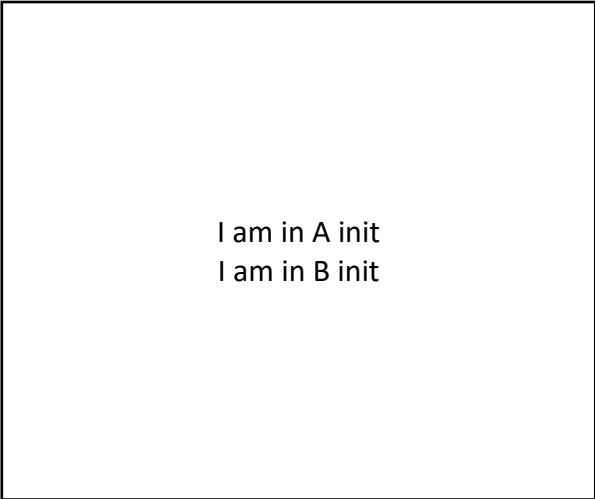
#### **Output:**

```
I teach C programming
I work in Subhash Programming Classes
I teach C++ programming
I work in Subhash Programming Classes
```

```
c = CTeacher()  
c.teach()  
c.schoolname()  
  
cpp = CppTeacher()  
cpp.teach()  
cpp.schoolname()
```

**Program 3:**

```
class A:  
    def __init__(self):  
        print("I am in A init")  
  
class B(A):  
    def __init__(self):  
        print("I am in B init")  
  
a = A()  
b = B()
```

**Output:**

I am in A init  
I am in B init

**Program 4:**

```
class A:  
    def __init__(self):  
        print("I am in A init")  
  
class B(A):  
    def __init__(self):  
        print("I am in B init")  
  
b = B()
```

**Output:**

I am in B init

**Program 5:**

```
class A:
    def __init__(self):
        print("I am in A init")

class B(A):
    def __init__(self):
        print("I am in B init")
        super().__init__()

b = B()
```

**Output:**

```
I am in B init
I am in A init
```

**Program 6:**

```
class A:
    a = 50
    def __init__(self):
        print("I am in A init")
        self.a = 10
        self.b = 20

class B(A):
    b = 30
    def __init__(self):
        print("I am in B init")
        super().__init__()

    def getValue(self):
        print("My value is = ", self.a)
        print("My value is = ", self.b)
        print("My value is = ", B.b)

b = B()
b.getValue()
```

**Output:**

```
I am in B init
I am in A init
My value is = 10
My value is = 20
My value is = 30
```

**Program 7:**

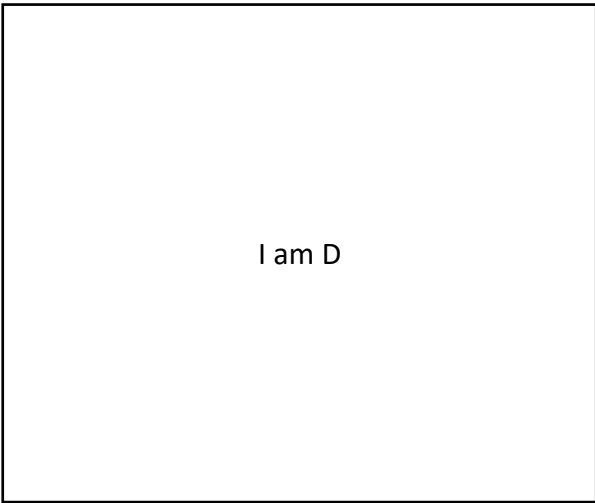
```
class A:
    def call(self):
        print("I am A")

class B(A):
    def call(self):
        print("I am B")

class C(A):
    def call(self):
        print("I am C")

class D(B,C):
    def call(self):
        print("I am D")

d = D()
d.call()
```

**Output:**

I am D

**Program 8:**

```
class A:
    def call(self):
        print("I am A")

class B(A):
    def call(self):
        print("I am B")

class C(A):
    def call(self):
        print("I am C")

class D(B,C):
    pass

d = D()
d.call()
```

**Output:**

I am B

**Program 9:**

```
class A:
    def call(self):
        print("I am A")

class B(A):
    def call(self):
        print("I am B")

class C(A):
    def call(self):
        print("I am C")

class D(C,B):
    pass

d = D()
d.call()
```

**Output:**

I am C

**Program 10:**

```
class A:
    def call(self):
        print("I am A")

class B(A):
    pass

class C(A):
    def call(self):
        print("I am C")

class D(B,C):
    pass

d = D()
d.call()
```

**Output:**

I am C

**Program 11:**

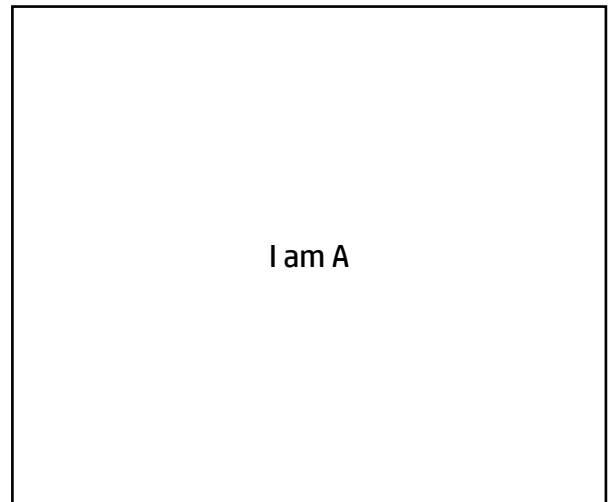
```
class A:
    def call(self):
        print("I am A")

class B(A):
    pass

class C(A):
    pass

class D(B,C):
    pass

d = D()
d.call()
```

**Output:**

I am A

**Program 12:**

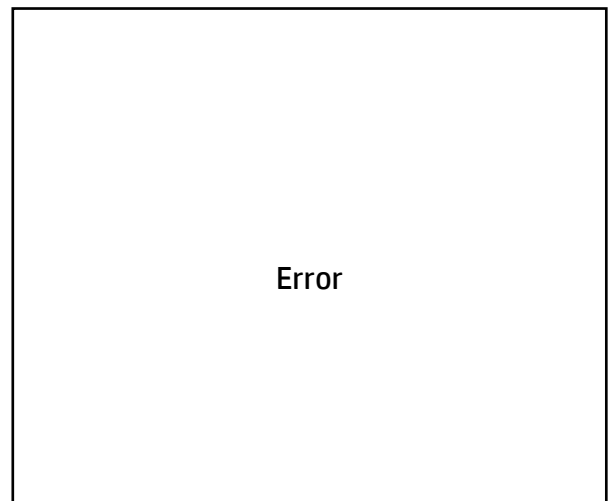
```
class A:
    pass:

class B(A):
    pass

class C(A):
    pass

class D(B,C):
    pass

d = D()
d.call()
```

**Output:**

Error

**Program 13:**

```
class A:  
    pass  
  
class B(A):  
    pass  
  
class C(A):  
    pass  
  
class D(B,C):  
    pass  
  
class E(A):  
    def call(self):  
        print("I am E")  
  
d = D()  
d.call()
```

**Output:**

Error

**Program 14:**

```
class A:  
    pass  
  
class B(A):  
    pass  
  
class C(A):  
    pass  
  
class D(B,C):  
    pass  
  
class E(A):  
    def call(self):  
        print("I am E")  
  
d = D()  
d.call()
```

**Output:**

Error

**Program 15:**

```
class A:
    def call(self):
        print("I am A")

class B(A):
    pass

class C(A):
    pass

class E(A):
    def call(self):
        print("I am E")

class D(B,C,E):
    pass

d = D()
d.call()
```

**Output:**

I am E

**Program 16:**

```
class A:
    def call(self):
        print("I am A")

class B(A):
    def call(self):
        print("I am B")

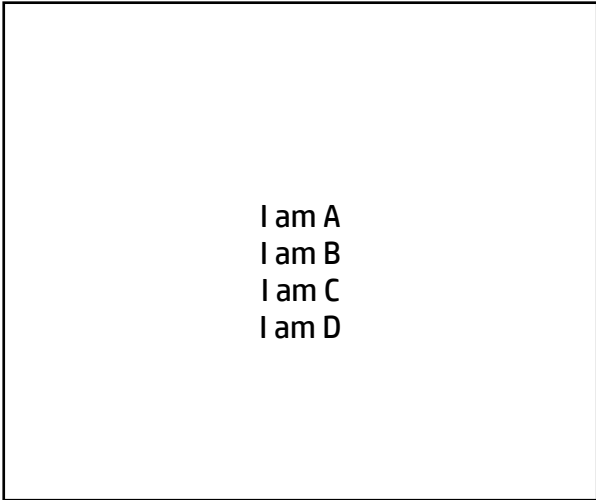
class C(A):
    def call(self):
        print("I am C")

class D(B,C):
    def call(self):
        print("I am D")

def jam(x):
    x.call()

a = A()
b = B()
c = C()
d = D()

jam(a)
jam(b)
jam(c)
jam(d)
```

**Output:**

I am A  
I am B  
I am C  
I am D



**Abstract Classes & Abstract Methods****Program 17:**

```
from abc import *

class Teacher(ABC):

    def schoolname(self):
        print("I teach at Subhash Programming Classes")

    @abstractmethod
    def teach(self):
        pass

class CTeacher(Teacher):

    def teach(self):
        print("I teach C")

class PythonTeacher(Teacher):

    def teach(self):
        print("I teach Python")

t = CTeacher()
t.schoolname()
t.teach()

print()

t = PythonTeacher()
t.schoolname()
t.teach()
```

**Output:**

```
I teach at Subhash Programming Classes
I teach C
```

```
I teach at Subhash Programming Classes
I teach Python
```

**Program 18:**

```
from abc import *  
  
class Teacher(ABC):  
    def schoolname(self):  
        print("I teach at Subhash Programming Classes")  
  
    @abstractmethod  
    def teach(self):  
        pass  
  
class CTeacher(Teacher):  
    def teach(self):  
        print("I teach C")  
  
class pythonTeacher(Teacher):  
    def teach(self):  
        print("I teach Python")  
  
t = Teacher()  
t.schoolname()
```

**Output:**

Error

**Program 19:**

```
from abc import *  
  
class Teacher(ABC):  
    def schoolname(self):  
        print("I teach at Subhash Programming Classes")  
  
    @abstractmethod  
    def teach(self):  
        pass  
  
class CTeacher(Teacher):  
    pass  
  
class PythonTeacher(Teacher):  
    pass  
  
t = CTeacher()  
t = PythonTeacher()
```

**Output:**

Error

**Demonstrating Polymorphism:****Program 20:**

```
from abc import *

class Shape(ABC):

    @abstractmethod
    def draw():
        pass

class Square(Shape):
    def draw(self):
        print("I am a square")

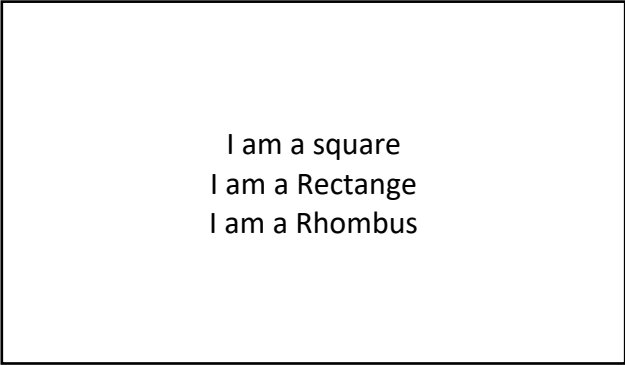
class Rectangle(Shape):
    def draw(self):
        print("I am a Rectangle")

class Rhombus(Shape):
    def draw(self):
        print("I am a Rhombus")

s = Square()
r = Rectangle()
rh = Rhombus()

shapes = [ s, r, rh ]

for i in shapes:
    i.draw()
```

**Output:**

```
I am a square
I am a Rectangle
I am a Rhombus
```