Accelerating Information Technology Innovation

http://aiti.mit.edu

Kenya Summer 2011
Lecture 07 – Inheritance

MISTI

Agenda

- Goal
  - Use objects to represent everyone in this course
  - Include:
    - Instructors
    - Students
    - Students who assist instructors
- Learn about
  - Inheritance
  - Multiple inheritance
  - Private Variables

```
class Participant:
    def __init__(self, name, university, status, grade, role):
        self.name = name
        self.university = university
        self.status = status
        self.grade = grade
        self.role = role
    def identify(self):
        print 'Name:%s \tAffiliation:%s \tStatus:%s \tGrade:%d \tRole:%s' %
   (self.name, self.university, self.status, self.grade, self.role)

>>> Student1 = Participant("John ","Strathmore","Student", "A", "Student")
>>> Student1.identify()
Name:John Mbithi    Uni:Strathmore  Status:Student  Grade: 1    Role:Student

>>> TechLead = Participant("Oshani","MIT","Instructor", "?","Tech Lead")
>>> TechLead.identify()
Name:Oshani      Uni:MIT    Status:Instructor   Grade:?     Role:Tech Lead
```

But we have extra fields we do not always need

```
class Student:
    def __init__(self, name, university, status, grade):
        self.name = name
        self.university = university
        self.status = status
        self.grade = grade
    def identify(self):
        print 'Name:%s \tUni:%s \tStatus:%s \tGrade:%d' %
   (self.name, self.university, self.status, self.grade)

class Instructor:
    def __init__(self, name, university, status, role):
        self.name = name
        self.university = university
        self.status = status
        self.role = role
    def identify(self):
        print 'Name:%s \tUni:%s \tStatus:%s \tRole:%s' %
   (self.name, self.university, self.status, self.role)
```

But wait, don't they share similar parts?

What's wrong with both approaches?

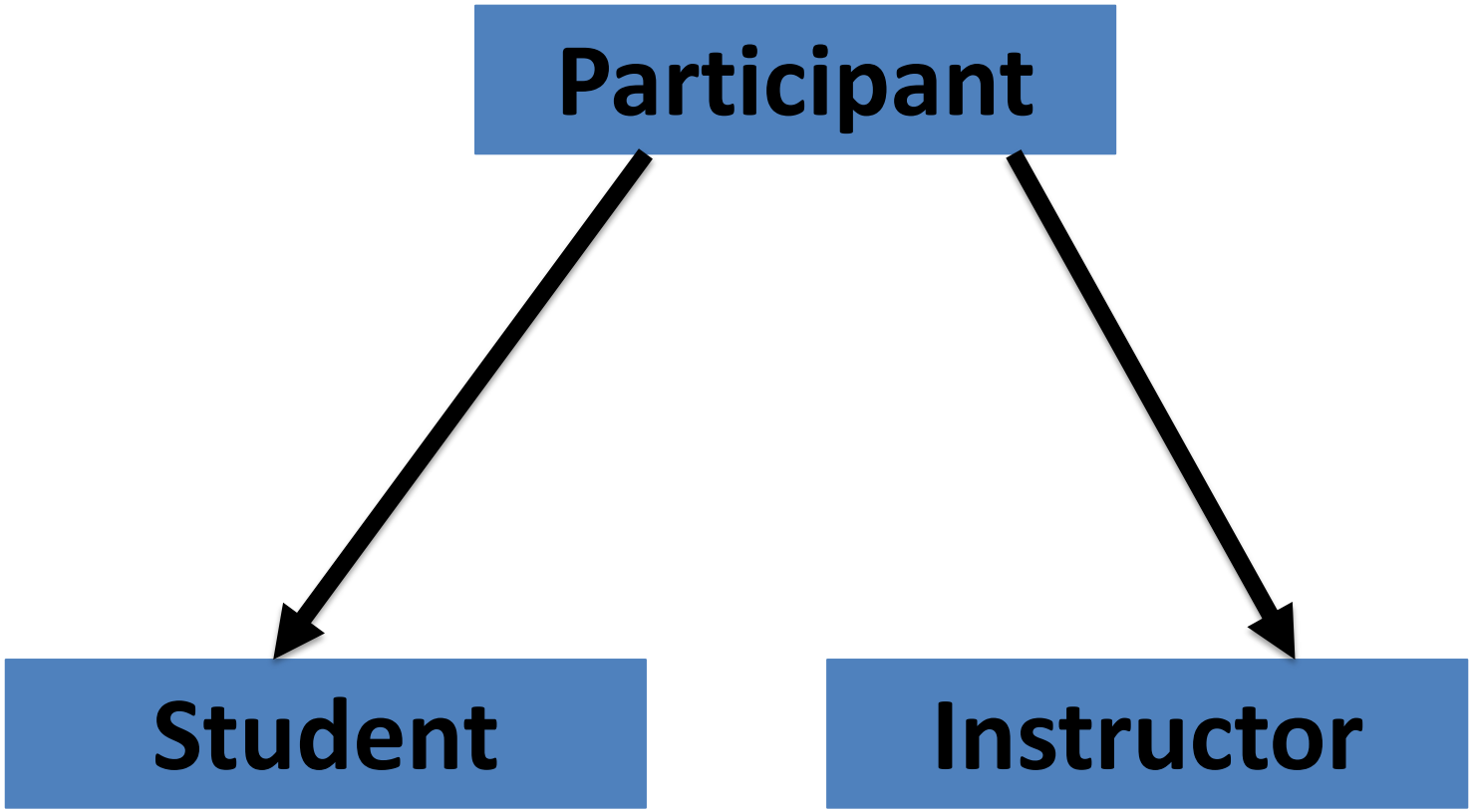
One class

- Sometimes variables are irrelevant
- A very large, complex class

Two classes

- Repeated behavior
- Increases our work
- We might not implement shared features the same way in the two classes

Inheritance



```
class Participant:
    def __init__(self, name, university, status):
        self.name = name
        self.university = university
        self.status = status
    def identify(self):
        print 'Name:%s \tUni:%s \tStatus:%s' % (self.name, self.university,
   self.status),

class Student(Participant):
    def __init__(self, name, university, grade):
        Participant.__init__(self, name, university, "Student")
        self.grade = grade
    def identify(self):
        Participant.identify(self)
        print '\tGroup:%d' % self.group

class Instructor(Participant):
    def __init__(self, name, university, role):
        Participant.__init__(self, name, university, "Instructor")
        self.role = role
    def identify(self):
        Participant.identify(self)
        print '\tRole:%s' % self.role
```

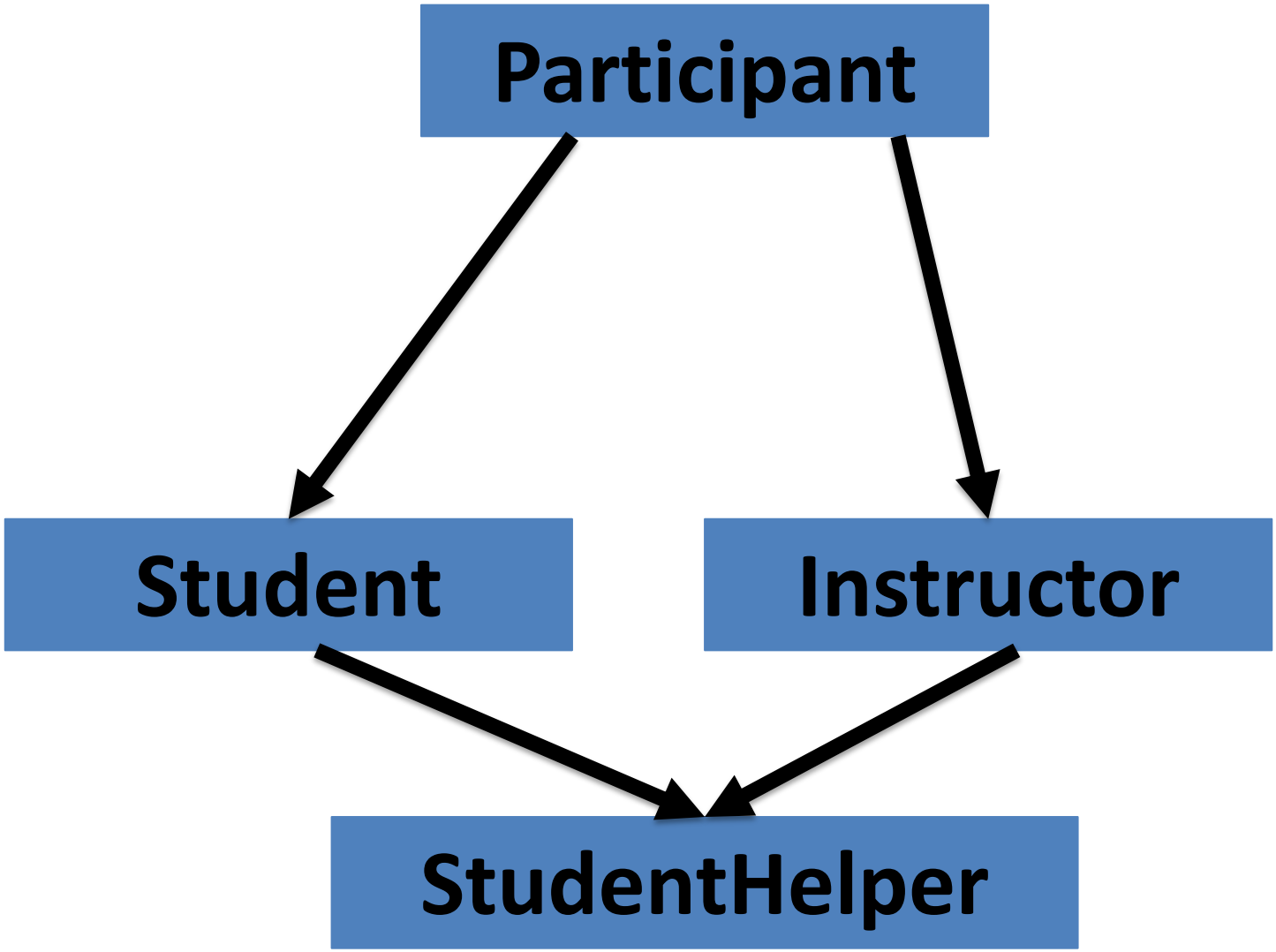
```
>>>Student1 = Student("John Mbithi","Strathmore", "A")
>>> Student1.identify()
Name:John Mbithi   Uni: Strathmore    Status:Student     Grade:A

>>> TechLead = Instructor("Oshani","MIT","Tech Lead")
>>> TechLead.identify()
Name:Oshani    Uni:MIT  Status:Instructor    Role:Tech Lead
```

Agenda

- Goal
  - Use objects to represent everyone in this course
  - Include:
    - Instructors
    - Students
    - **Students who assist instructors**
- Learn about:
  - Inheritance
  - Multiple inheritance
  - Private Variables

Multiple Inheritance



```
class Participant:
    def __init__(self, name, university, status):
        self.name = name
        self.university = university
        self.status = status
    def identify(self):
        print 'Name:%s \tUni:%s \tStatus:%s' % (self.name, self.university,
   self.status),

class Student(Participant):
    def __init__(self, name, university, grade):
        Participant.__init__(self, name, university, "Student")
        self.grade = grade
    def identify(self):
        Participant.identify(self)
        print '\tGrade:%d' % self.group

class Instructor(Participant):
    def __init__(self, name, university, role):
        Participant.__init__(self, name, university, "Instructor")
        self.role = role
    def identify(self):
        Participant.identify(self)
        print '\tRole:%s' % self.role
```

```
class HelperStudent(Student, Instructor):
    def __init__(self, name, university, group):
        Instructor.__init__(self, name, university, "Helper")
        Student.__init__(self, name, university, group)

>>> HelpStd1 = HelperStudent("Kelvin","Strathmore", "A")

>>> HelpStd1.identify()
Name:Kelvin    Uni:Strathmore     Status:Student     Grade:A

>>> print HelpStd1.role
Helper
```

Private Variables

- In Python, our class variables are “exposed”
- How do we keep variables hidden from outsiders

```
class HelperStudent(Student, Instructor):
    def __init__(self, name, university, grade, email):
        Instructor.__init__(self, name, university, "Helper")
        Student.__init__(self, name, university, grade)
        self.__email = email
    def getEmail(self):
        return self.__email

>>> HelpStd1 = HelperStudent("Kelvin","Strathmore", "A","kelvin@yahoo.com")

>>> HelpStd1.identify()
Name:Kelvin Uni:Strathmore  Status:Student  Grade:A

>>> print HelpStd1.role
Helper

>>> print HelpStd1.__email
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    print HelpStd1.__email
AttributeError: HelperStudent instance has no attribute '__email'

>>> print HelpStd1.getEmail()
kelvin@yahoo.com
```