Method Overriding

By the end of this video you will be able to…

- Create methods which override from a superclass
- Contrast method overloading and method overriding

What did we want?

1. Keep common behavior in one class
2. Split different behavior into separate classes
3. Keep all of the objects in a single data structure

Overloading vs. Overriding

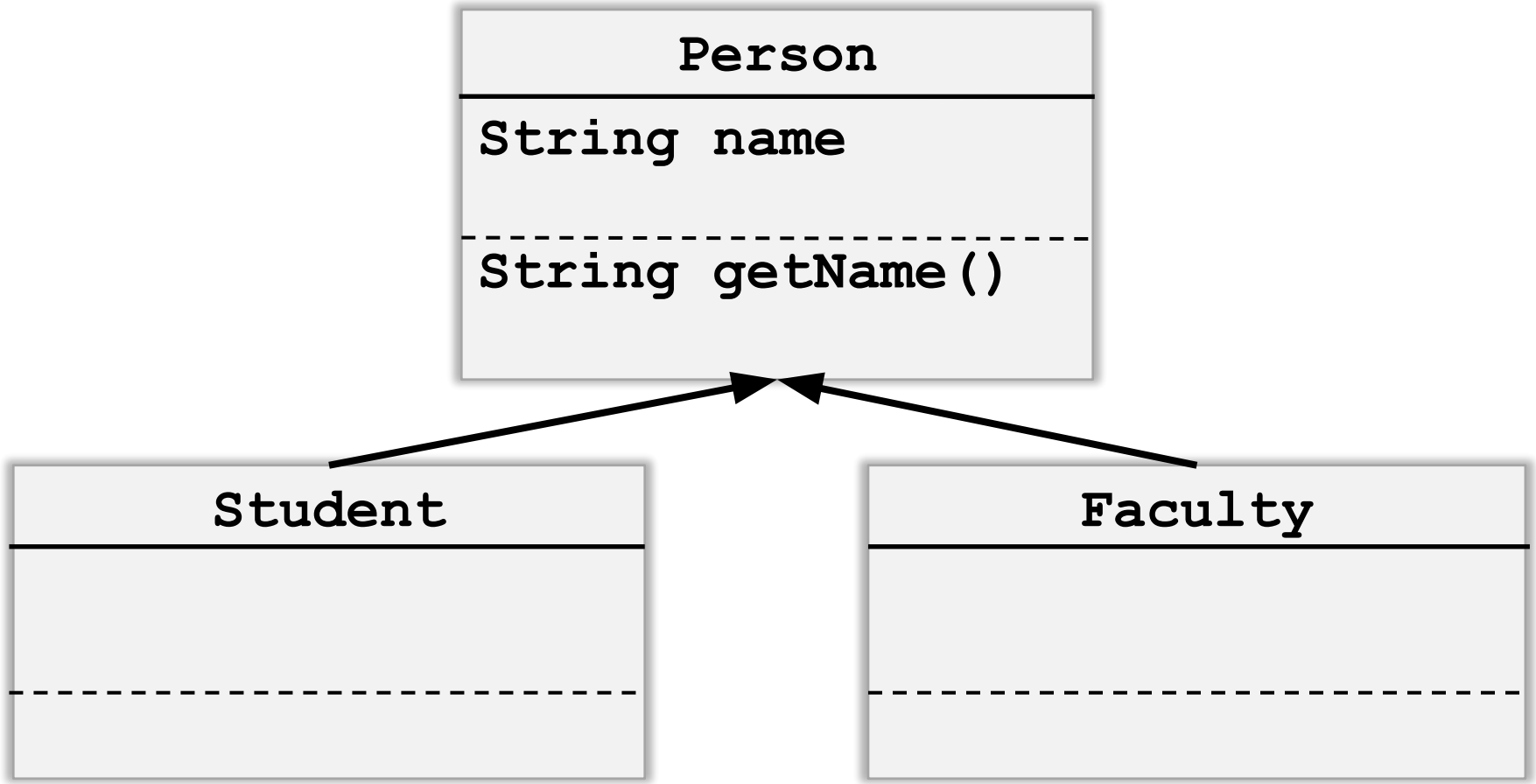
- **Overloading**: Same class has same method name with different parameters

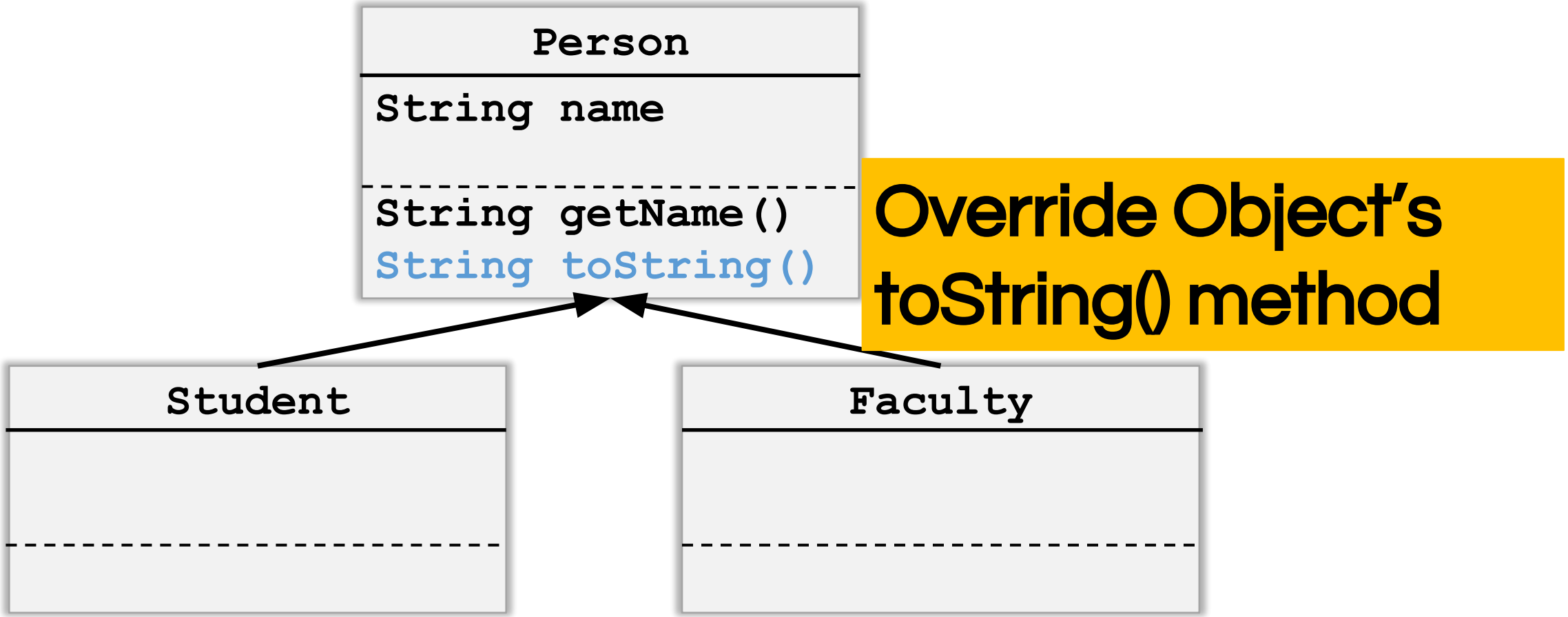
- **Overriding**: Subclass has same method name with the same parameters as the superclass

Object class

Modifier and Type	Method and Description
String	toString() Returns a string representation of the object.

http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html

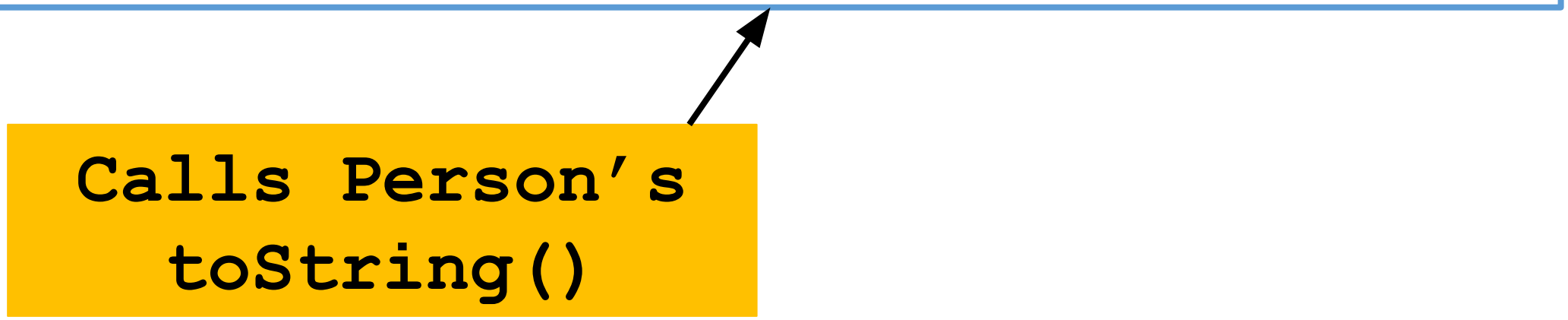

Person
String name
String getName()
Student
Faculty


Person
String name
String getName()
String toString()
Student
Faculty
Override Object's
toString() method

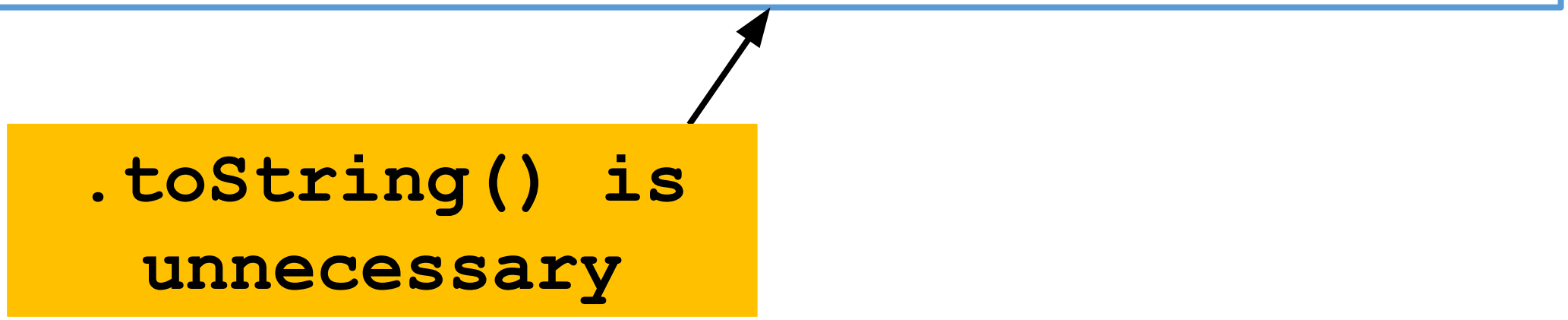
```
public class Person {
 private String name;

 public String toString() {
    return this.getName();
 }
}
```

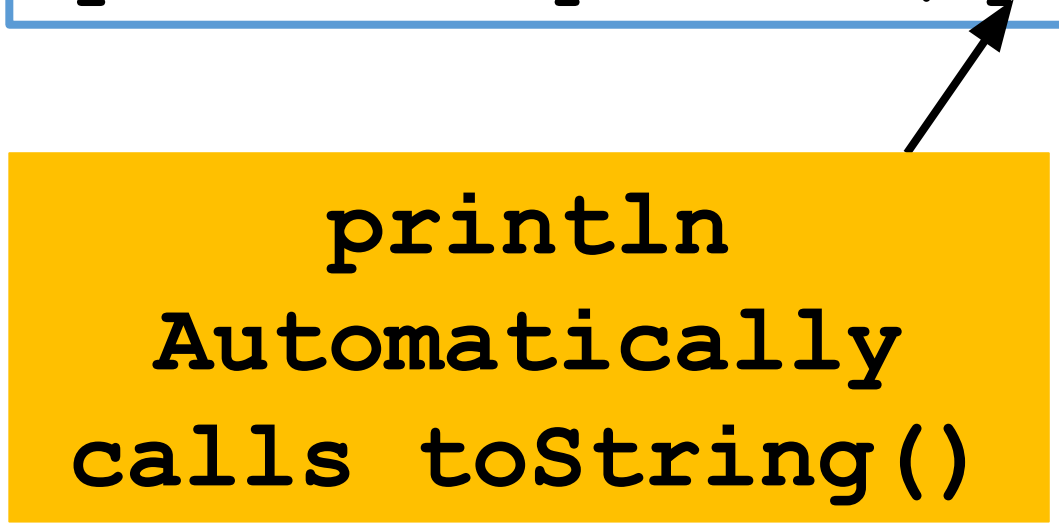
```
// assume ctor
Person p = new Person("Tim");
System.out.println( p.toString() );
```



```
// assume ctor
Person p = new Person("Tim");
System.out.println( p.toString() );
```



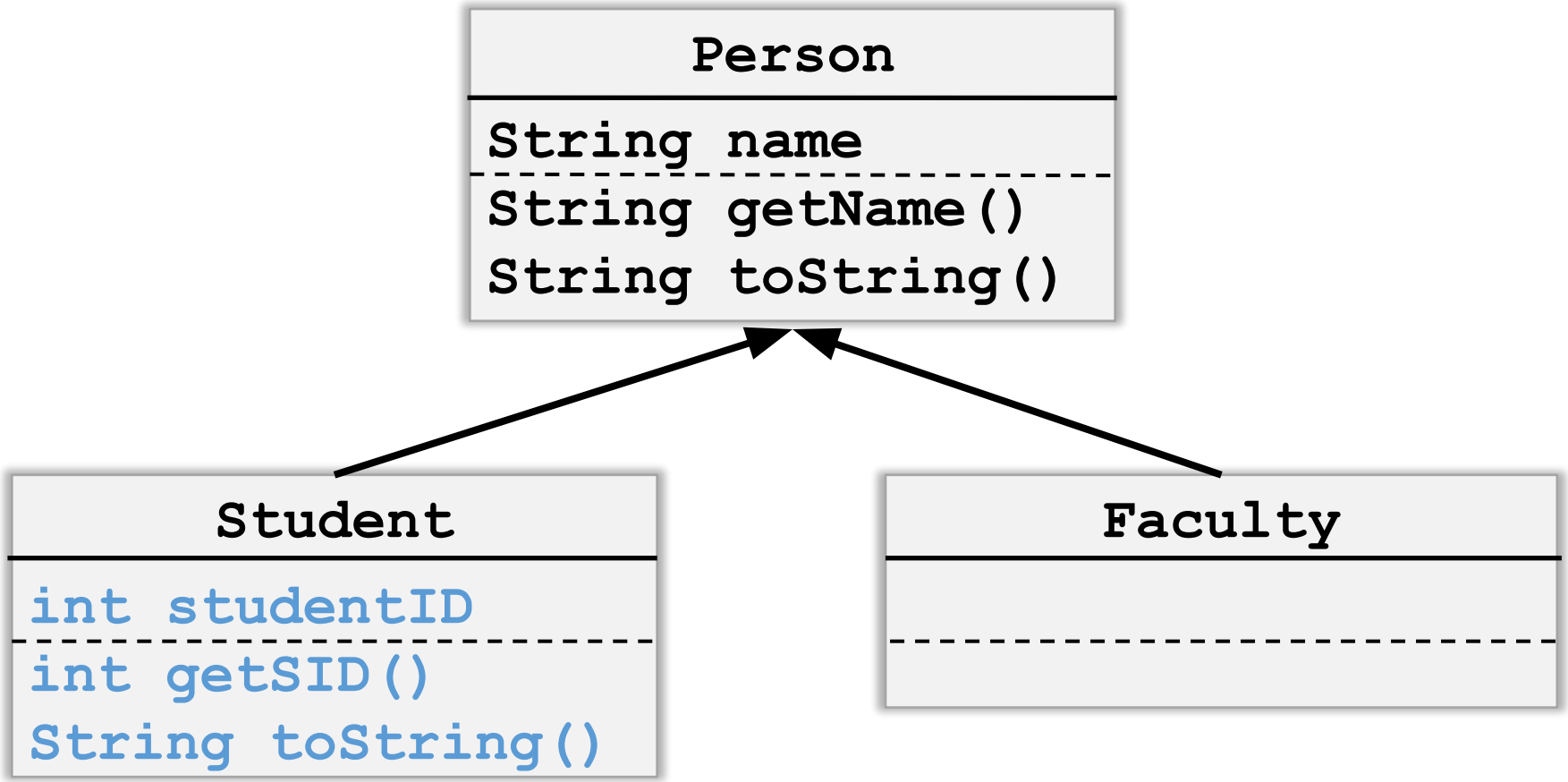
```
// assume ctor
Person p = new Person("Tim");
System.out.println( p );
```



```
// assume ctor
Person p = new Person("Tim");
System.out.println( p );
```

Output

```
Tim
```


Person
String name
String getName()
String toString()
Student
int studentID
int getSID()
String toString()
Faculty

```
public class Student extends Person {
 private int studentID;

 public int getSID() {
  return studentID;
 }


}
```

```
public class Student extends Person {
 private int studentID;

 public int getSID() {
  return studentID;
 }


}
```

Goal:

SID: Person info

```
public class Student extends Person {
 private int studentID;

 public int getSID() {
  return studentID;
 }

 public String toString() {
   return this.getSID() + ": " +
          this.getName();
 }
}
```

```
public class Student extends Person {
 private int studentID;

 public int getSID() {
  return studentID;
 }

 public String toString() {
   return this.getSID() + ": " +
          this.getName();
 }
}
```

What if Person changes?

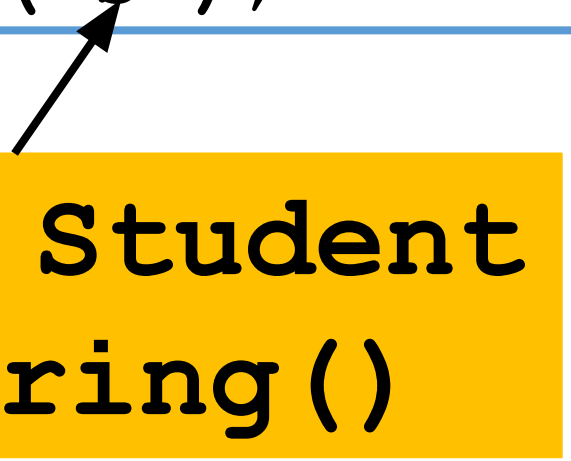
```
public class Student extends Person {
 private int studentID;

 public int getSID() {
  return studentID;
 }

 public String toString() {
   return this.getSID() + ": " +
          super.toString();
 }
}
```

"super" refers to superclass

```
// assume ctor
Student s = new Student("Cara",1234);
System.out.println( s );
```



```
// assume ctor
Student s = new Student("Cara",1234);
System.out.println( s );
```

Output

```
1234: Cara
```

```
// assume ctor
Person s = new Student("Cara",1234);
System.out.println( s );
```

```
// assume ctor
Person s = new Student("Cara",1234);
System.out.println( s );
```

Output

```
1234: Cara
```

```
// assume ctor
Person s = new Student("Cara",1234);
System.out.println( s );
```

Output

```
1234: Cara
```

Why? Polymorphism