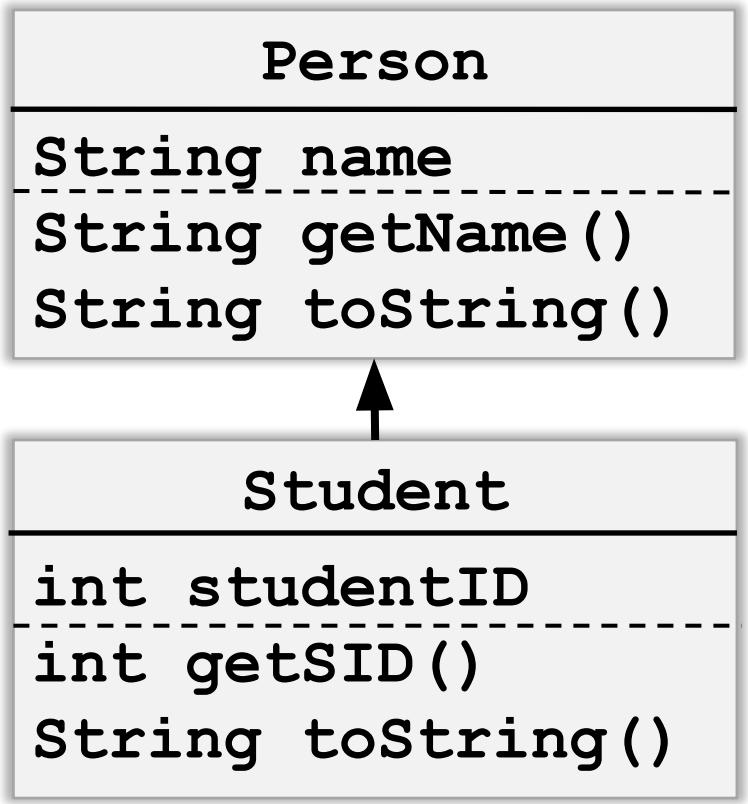
Polymorphism: Casting

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

- Step through decisions made at compile time and runtime
- Use casting of objects to aid the compiler




```
Person s = new Student("Cara",1234);
s.getSID();
```

Compile Time Error!

Casting

- Automatic type promotion (like `int` to `double`)
  - Superclass ref = new Subclass();

widening

Casting

- Automatic type promotion (like `int` to `double`)
  - Superclass superRef = new Subclass();
- Explicit casting (like `double` to `int`)
  - Subclass ref = (Subclass) superRef;

Casting

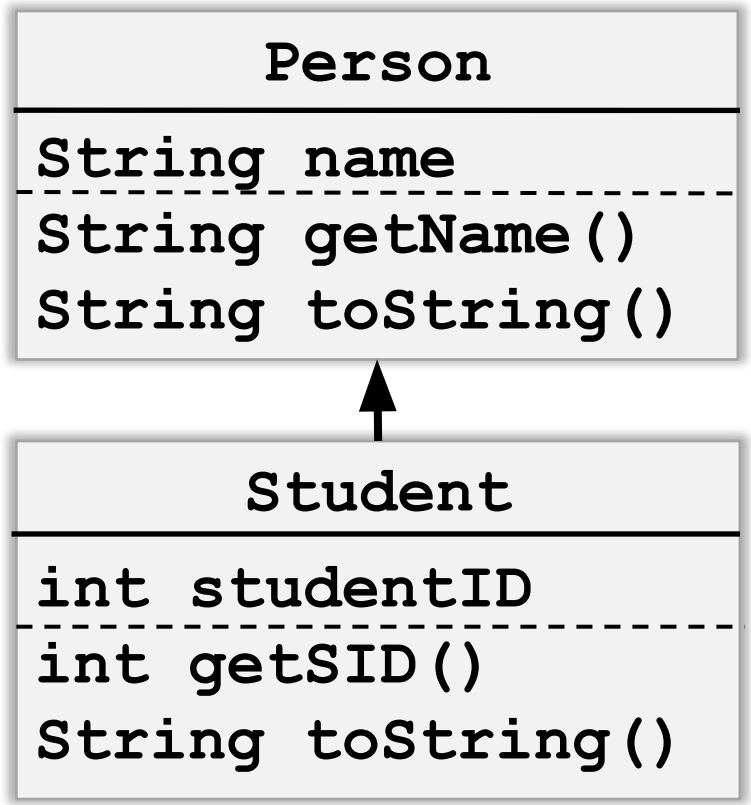
- Automatic type promotion (like `int` to `double`)
  - Superclass superRef = new Subclass();
- Explicit casting (like `double` to `int`)
  - Subclass ref = (Subclass) superRef;

narrowing

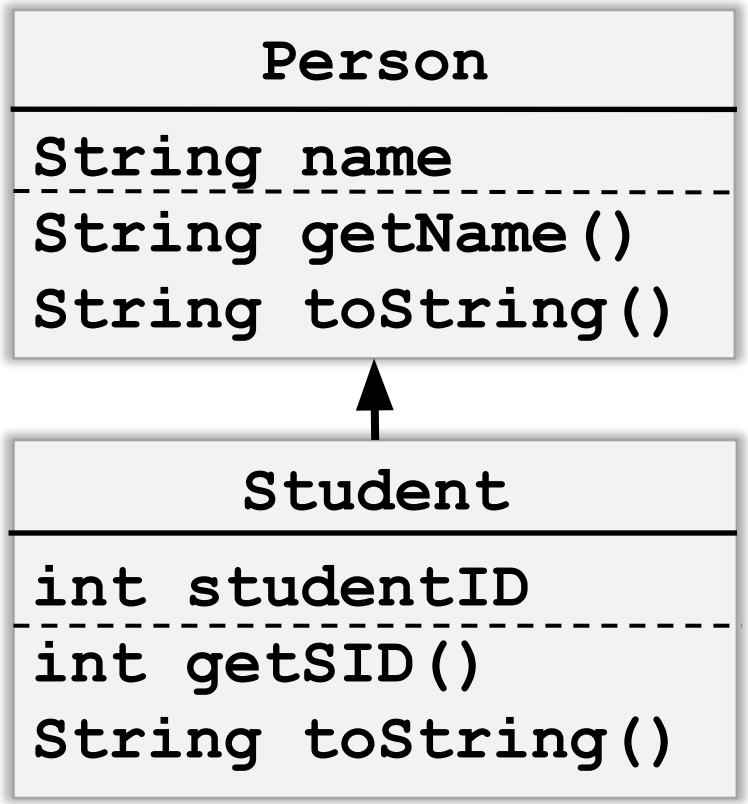
Casting

- Automatic type promotion (like `int` to `double`)
  - Superclass superRef = new Subclass();
- Explicit casting (like `double` to `int`)
  - Subclass ref = (Subclass) superRef;

BE CAREFUL:
Compiler trusts you

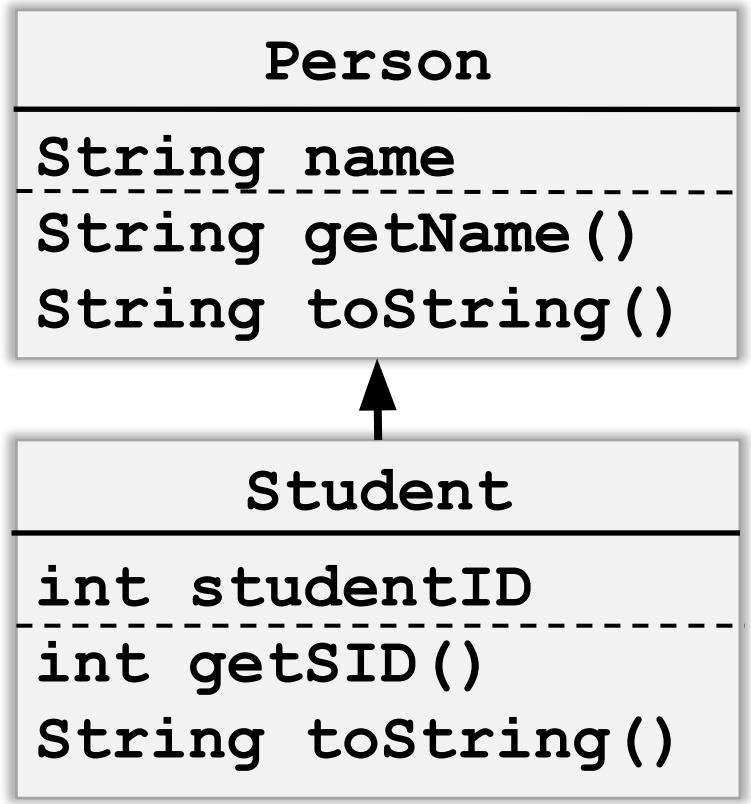



```
Person s = new Student("Cara",1234);
s.getSID();
```

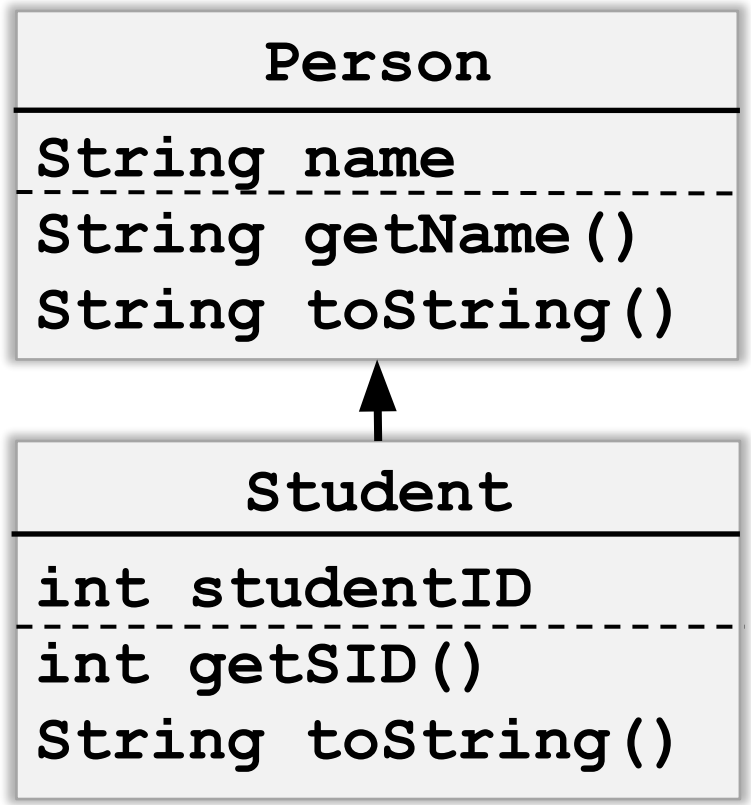



```
Person s = new Student("Cara",1234);
s.getSID();
( (Student)s ).getSID();
```

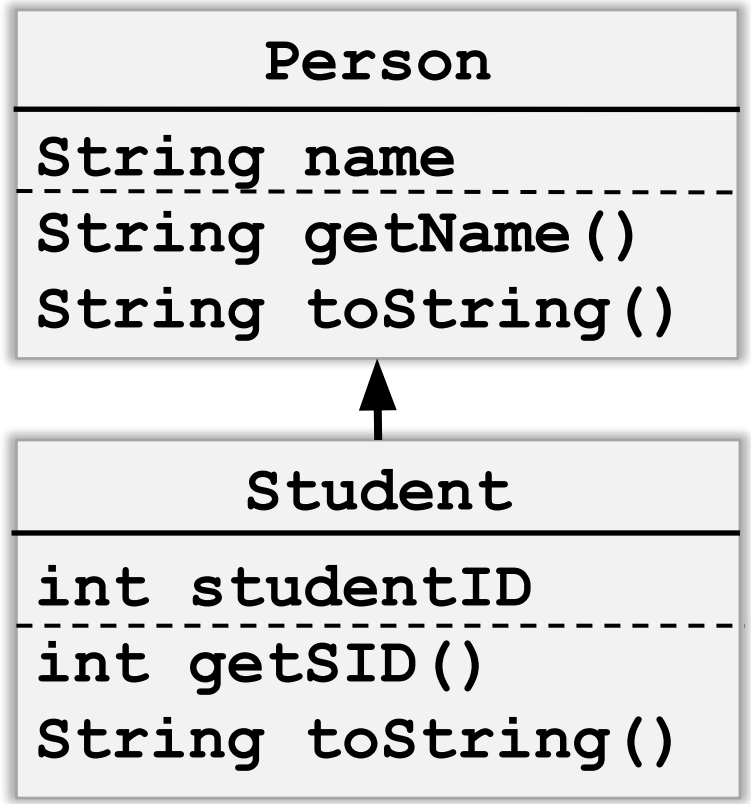
This works!



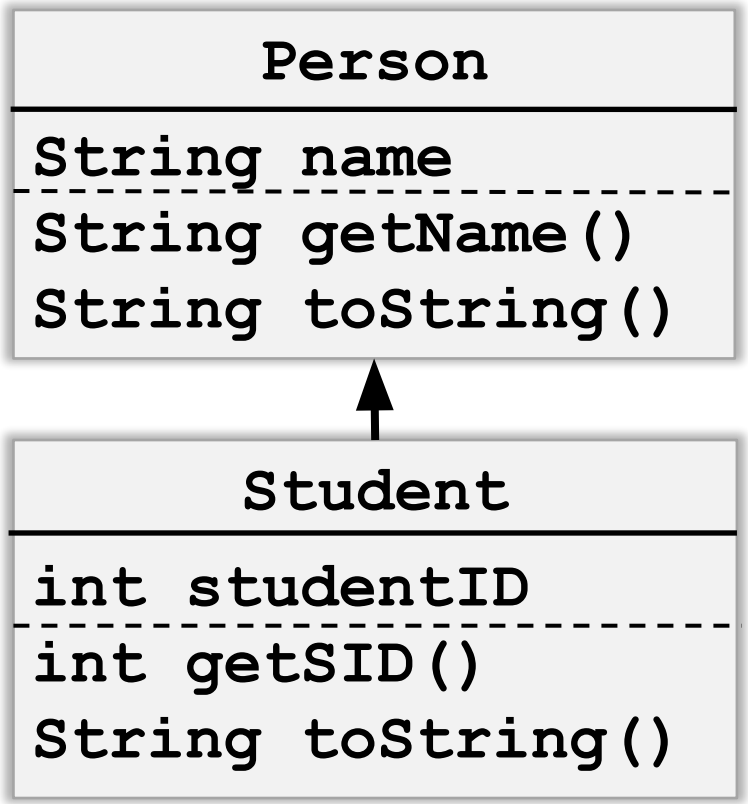

```
Person s = new Person("Tim");
( (Student)s ).getSID();
```




```
Person s = new Person("Tim");
( (Student)s ).getSID();
```




```
Person s = new Person("Tim");
( (Student)s ).getSID();
```




```
Person s = new Person("Tim");
( (Student)s ).getSID();
```

Runtime Error!

java.lang.ClassCastException: From Person to Student

Runtime type check

`instanceof`

- Provides runtime check of **is-a** relationship

Runtime type check

`instanceof`

- Provides runtime check of **is-a** relationship

```
if( s instanceof Student )
{
 // only executes if s is-a
 // Student at runtime
 ( (Student)s ).getSID();
}
```

Polymorphism

Compile Time Decisions

Runtime Decisions