

Polymorphism: Compile Time vs. Runtime



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/)
by Christine Alvarado, Mia Minnes, and Leo Porter, 2015.

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

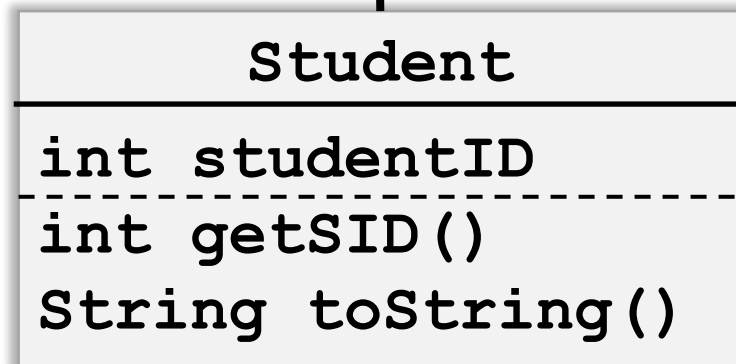
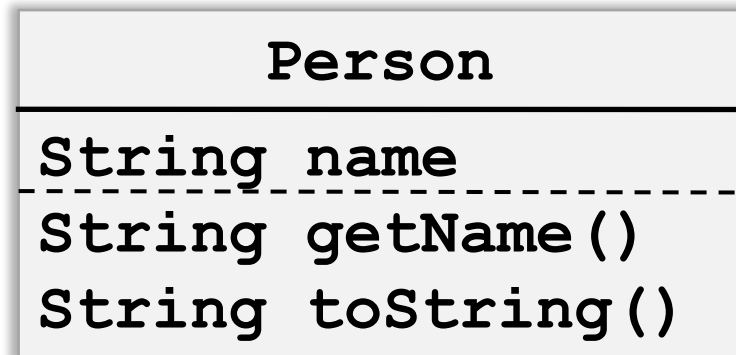
- Step through decisions made at compile time and runtime

Compile Time Rules

- Compiler ONLY knows reference type
- Can only look in reference type class for method
- Outputs a method signature

Compile Time Rules

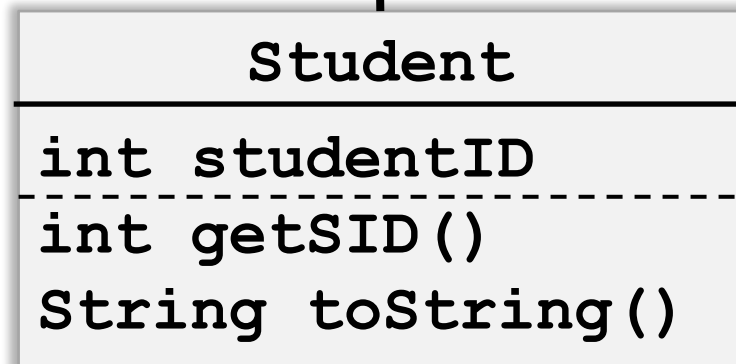
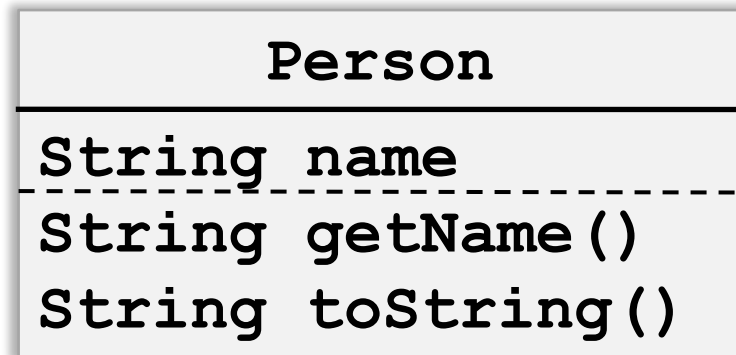
- Compiler ONLY knows reference type
- Can only look in reference type class for method
- Outputs a method signature



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

Compile Time Rules

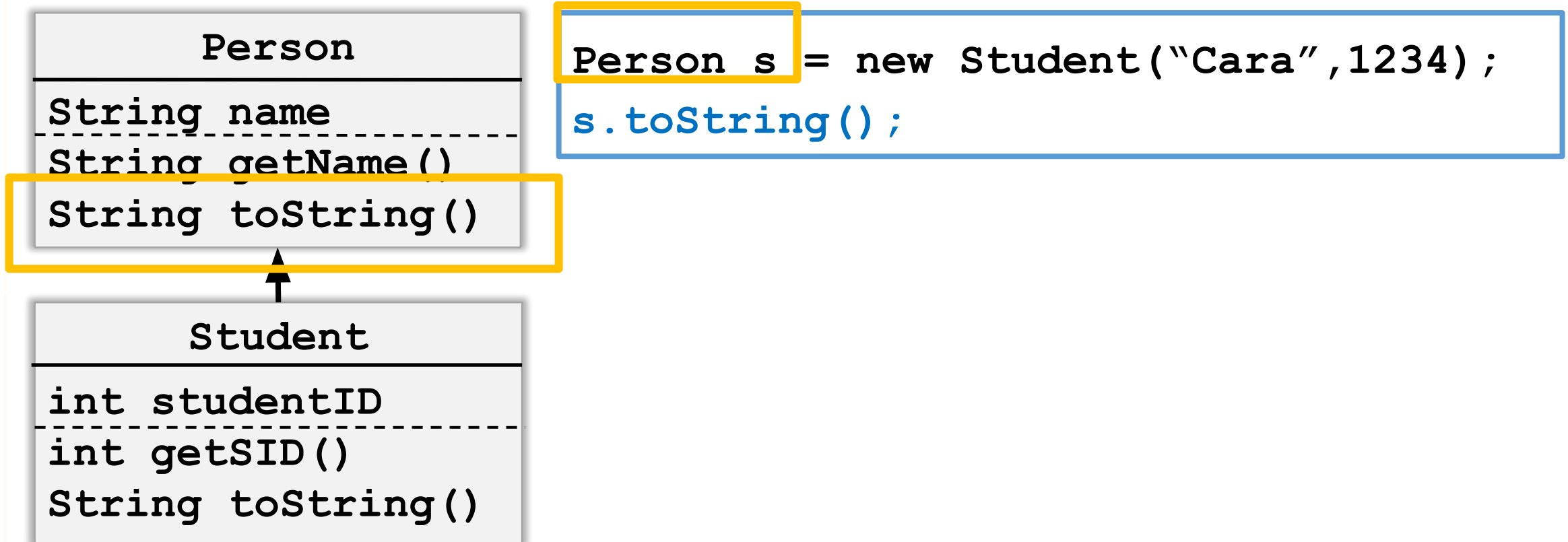
- Compiler ONLY knows reference type
- Can only look in reference type class for method
- Outputs a method signature



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

Compile Time Rules

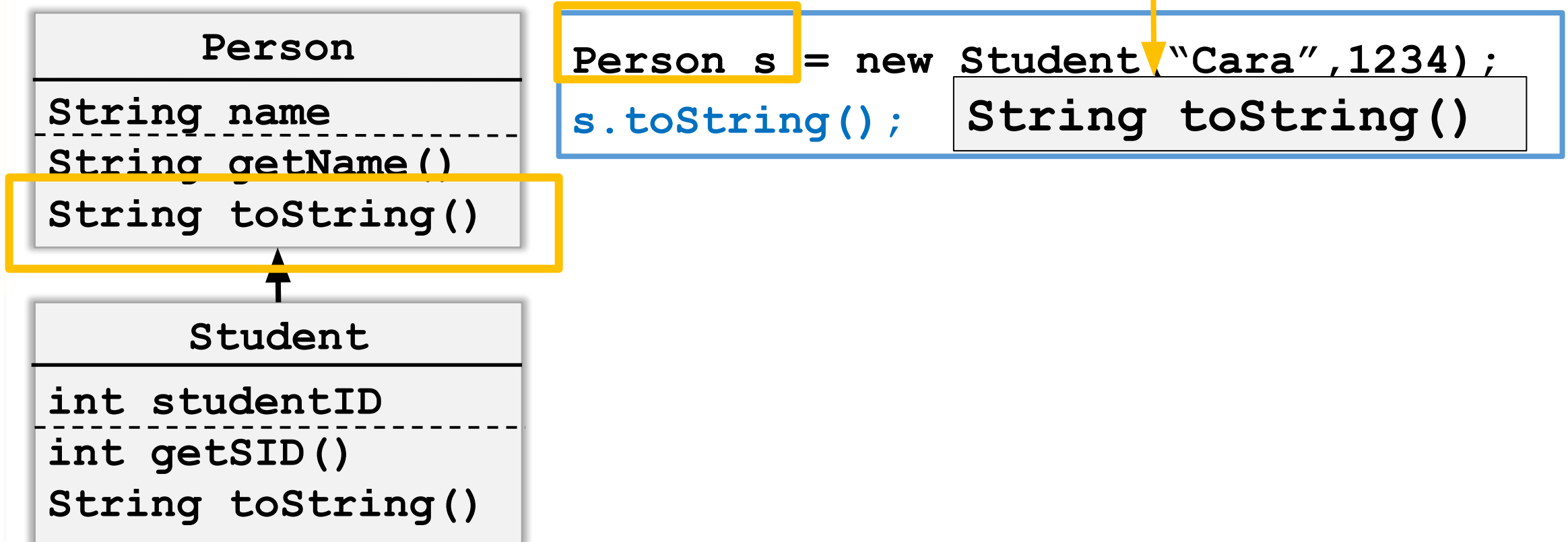
- Compiler ONLY knows reference type
- Can only look in reference type class for method
- Outputs a method signature



Compile Time Rules

- Compiler ONLY knows reference type
- Can only look in reference type class for method
- Outputs a method signature

Method Signature

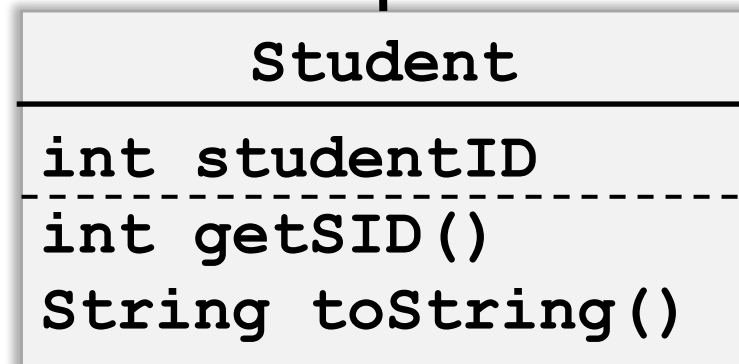
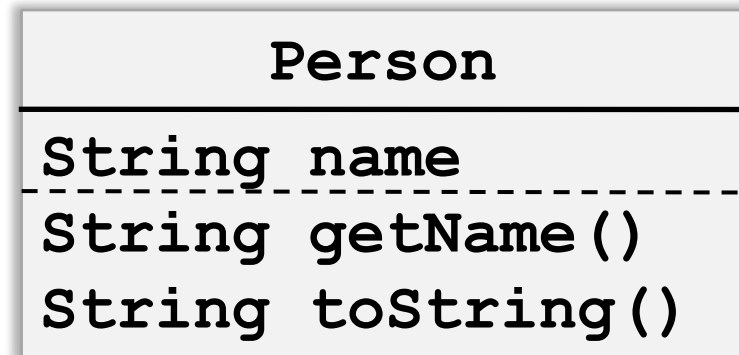


Run Time Rules

- Follow exact **runtime type** of object to find method
- Must match compile time method signature to appropriate method in actual object's class

Run Time Rules

- Follow exact **runtime type** of object to find method
- Must match compile time method signature to appropriate method in actual object's class

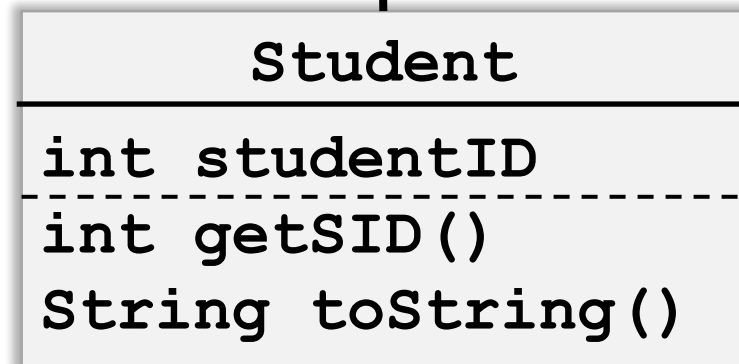
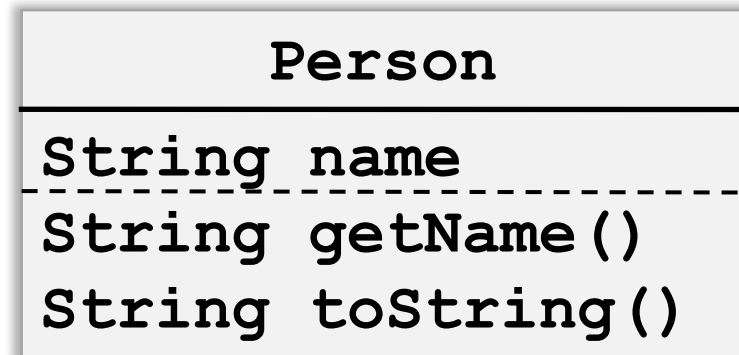


```
Person s = new Student("Cara", 1234);  
s.toString();
```

```
String toString()
```

Run Time Rules

- Follow exact **runtime type** of object to find method
- Must match compile time method signature to appropriate method in actual object's class

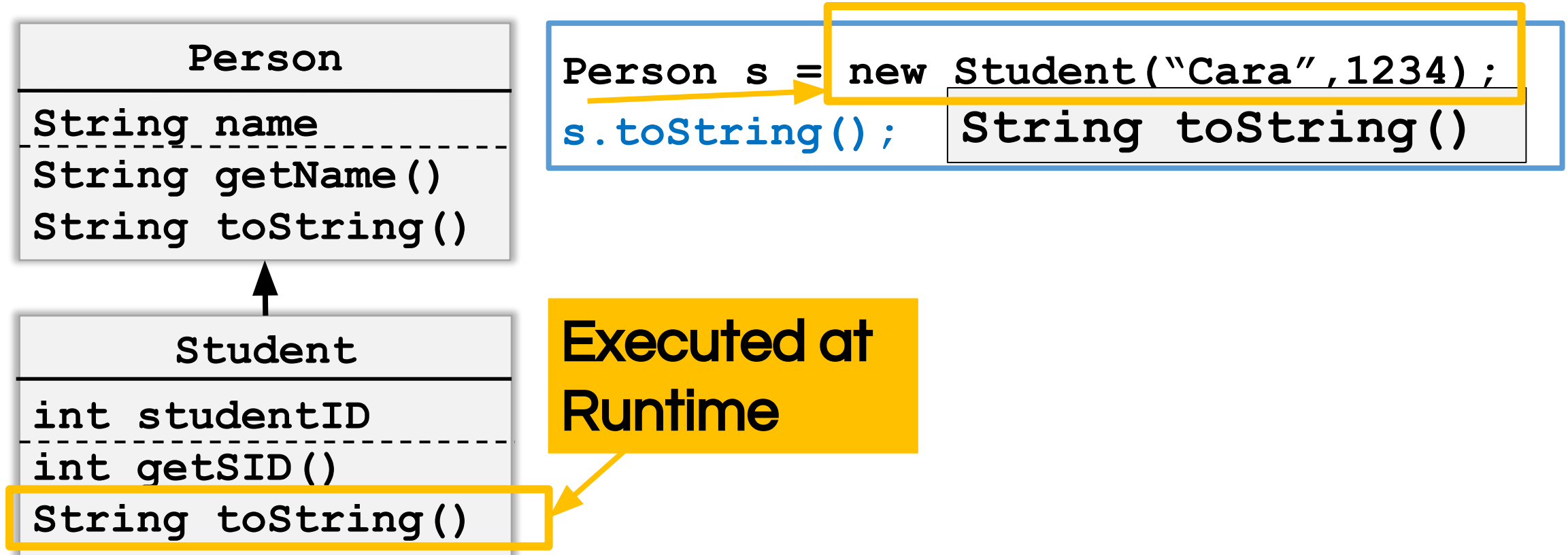


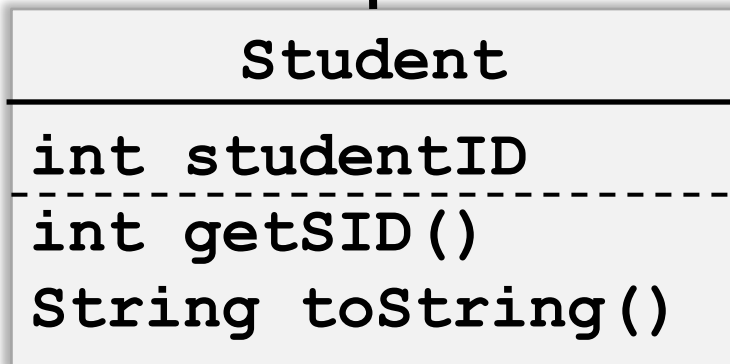
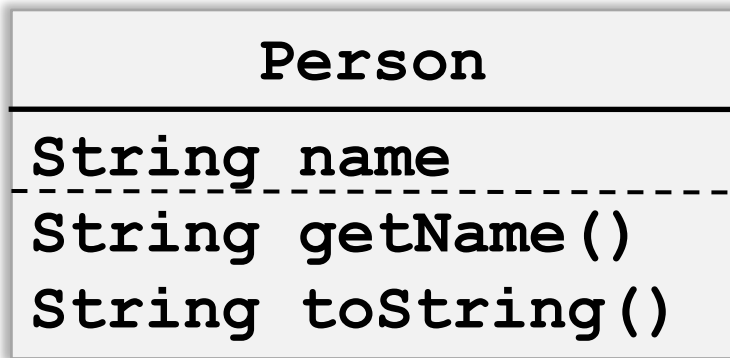
```
Person s = new Student("Cara", 1234);  
s.toString();
```

The diagram highlights the runtime type resolution process. A yellow box encloses the expression `new Student("Cara", 1234);`. An arrow points from the variable `s` to the `toString()` method call on the next line. A separate box shows the signature `String toString()`, which matches the one in the `Student` class, demonstrating that the runtime type (Student) determines the method to be executed.

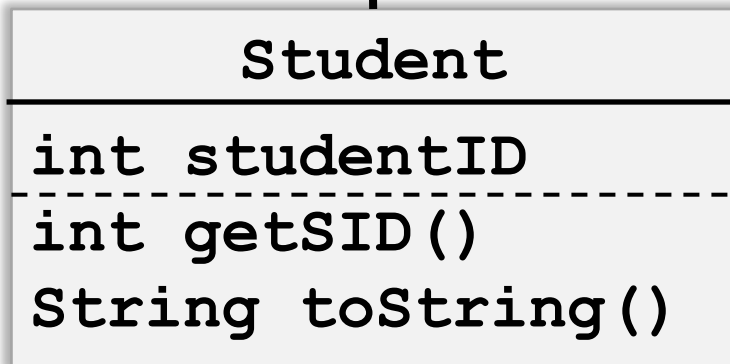
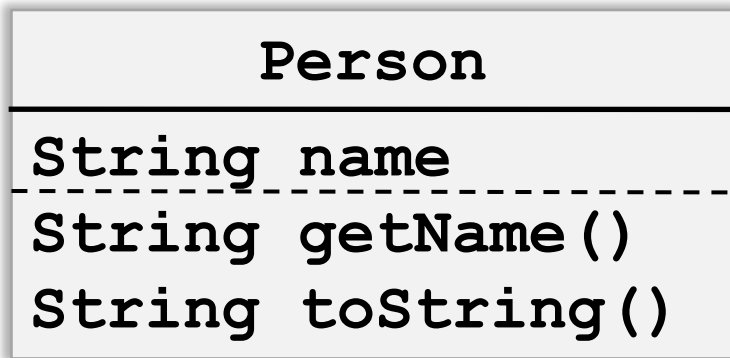
Run Time Rules

- Follow exact **runtime type** of object to find method
- Must match compile time method signature to appropriate method in actual object's class

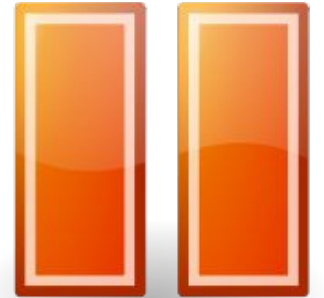


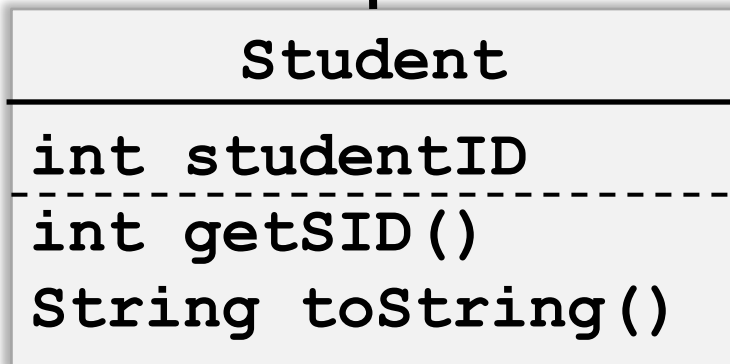
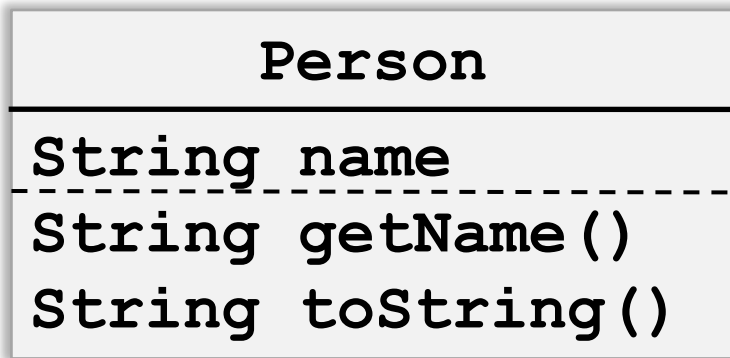


```
Person s = new Student("Cara", 1234) ;  
s.getSID() ;
```

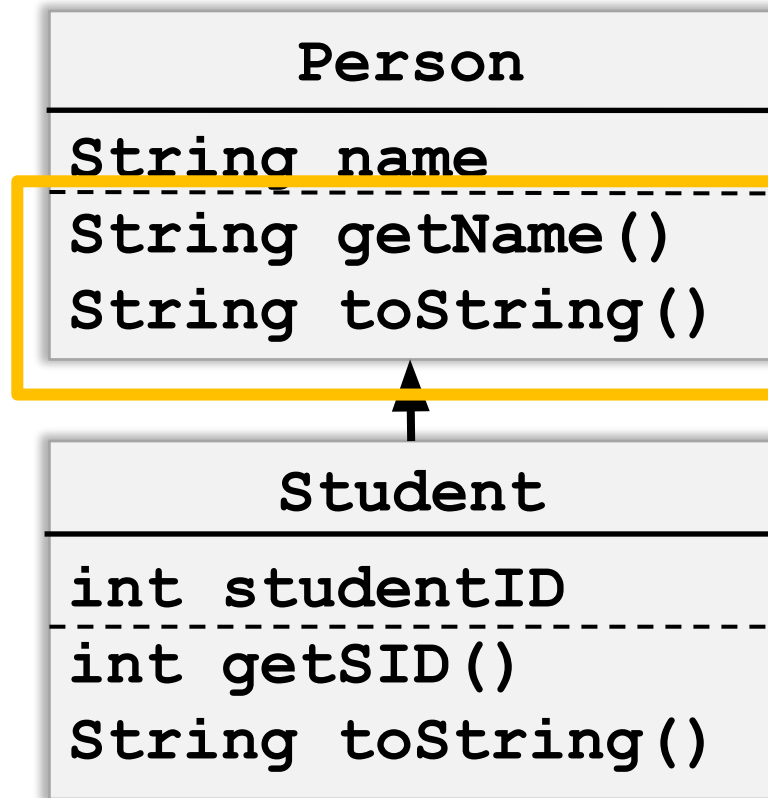


```
Person s = new Student("Cara", 1234) ;  
s.getSID() ;
```



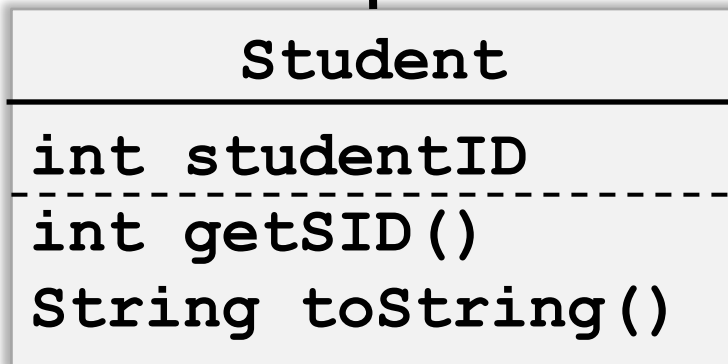
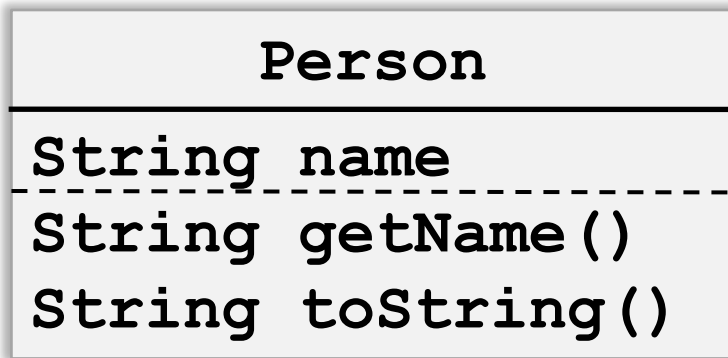


```
Person s = new Student("Cara", 1234) ;  
s.getSID() ;
```



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

**No getSID ()
method**



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

**Compile Time
Error!**