nt (Display 7.3). This means that an object of the class Undergraduate has ods and instance variables of the class Student. But Student is already a s of Person (Display 7.1). So this also means that an object of the class uate has all the methods and instance variables of the class Person. An class Person has the instance variable name. An object of the class Student nce variables name and studentNumber. An object of the class Undergrad- e instance variables name, studentNumber, and the added instance variable bject of the class Undergraduate must use accessor and mutator methods to hange the instance variables name and studentNumber, but it definitely has e variables.

.8 contains a UML diagram showing the relationship among the classes Per- nt, and Undergraduate.

a chain of derived classes like this can produce a good deal of efficient reus- Both the classes Undergraduate and Student (as well as any other classes either of them), in effect, reuse the code given in the definition of the class ause they inherit all the methods of the class Person.

constructors for the class Undergraduate. They each begin with an invoca- r, which in this context stands for a constructor of the base class Student. tructors for the class Student also begin with an invocation of super, which tands for a constructor of the base class Person. Thus, when a constructor for uate is invoked (using new), first a constructor for Person is invoked, then a or Student is invoked, and then all the code following super in the con- ndergraduate is executed.

e definition of the reset method in the class Undergraduate, which we re:

```
 void reset(String newName,
                     int newStudentNumber, int newLevel)

set(newName, newStudentNumber);
tLevel(newLevel); //Checks 1 <= newLevel <= 4
```

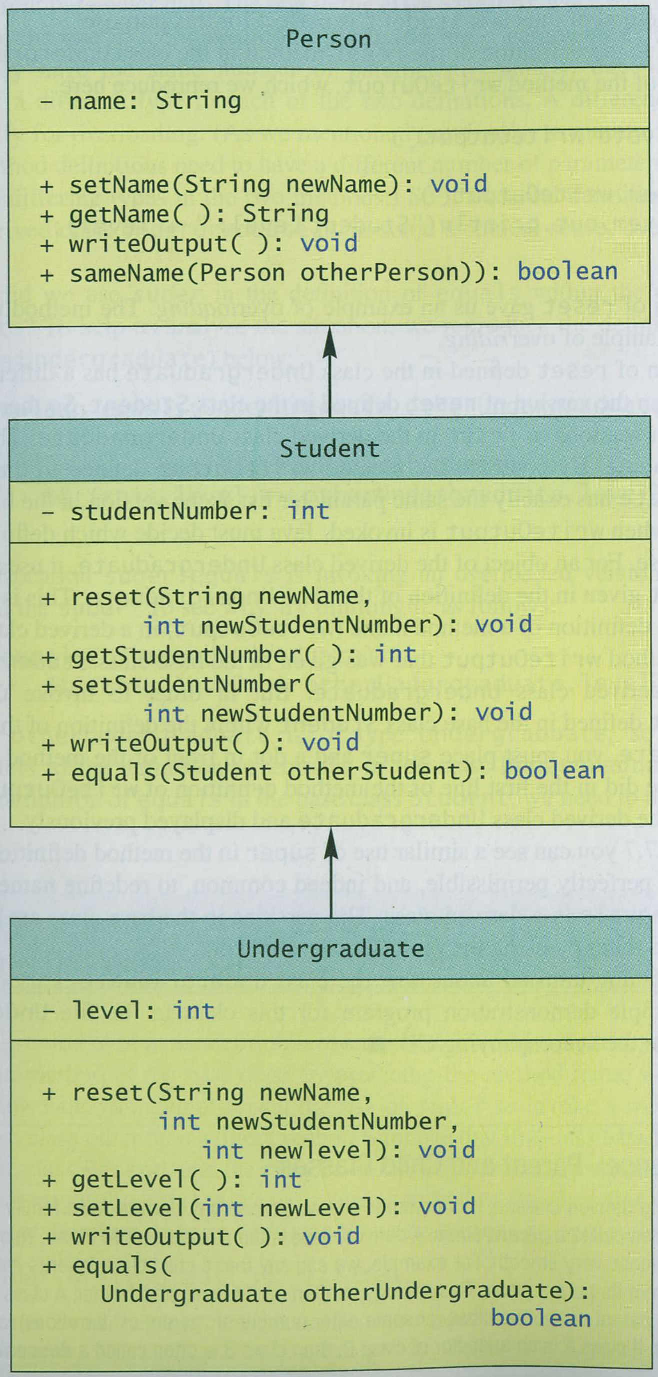
e method starts with an invocation of reset with only two arguments. s the method named reset in the base class Student. In the class ate, the method named reset is overloaded because there are two defi- he method name reset with different parameter lists. One takes two nd the other takes three arguments. The one that takes two arguments is m the class Student, but it is still a full-fledged method of the class ate.

e-argument version of reset (which is defined in the class Undergraduate ed in the previous displayed code), the method resets the values of the ables name and studentNumber (to newName and newStudentNumber, with the following invocation:

```
ewName, newStudentNumber);
```

instance variable level is reset to newLevel with a call to setLevel.

DISPLAY 7.8 Some More Details of a UML Class Hierarchy


Person
– name: String
+ setName(String newName): void
+ getName( ): String
+ writeOutput( ): void
+ sameName(Person otherPerson)): boolean
Student
– studentNumber: int
+ reset(String newName, int newStudentNumber): void
+ getStudentNumber( ): int
+ setStudentNumber(int newStudentNumber): void
+ writeOutput( ): void
+ equals(Student otherStudent): boolean
Undergraduate
– level: int
+ reset(String newName, int newStudentNumber, int newlevel): void
+ getLevel( ): int
+ setLevel(int newLevel): void
+ writeOutput( ): void
+ equals(Undergraduate otherUndergraduate): boolean