Object Oriented Programming – Java 1

Lecture 10

Polymorphism

Dr. Obuhuma James

Description

This topic explores polymorphism as one of the key features of object-oriented programming languages. Polymorphism means appearing in many forms. It only happens with the existence of multiple classes and inheritance between or among classes. Thus, this topic is anchored on earlier topics on Classes and Object, Inheritance, Encapsulation, and Abstraction already covered in the units.

Learning Outcomes

By the end of this topic, you will be able to:

- Describe the concept of polymorphism as a feature of the object-oriented programming style.
- Practically demonstrate the concept of polymorphism as applied in object-oriented programming.
- Differentiate between method overriding and method overloading.

Overview of Polymorphism

Among the key characteristics of object-oriented programming languages is polymorphism. So far, other characteristics already covered are classes, objects, inheritance, and encapsulation. Polymorphism means “many forms”. It happens whenever we have many classes that are related through inheritance. Thus, polymorphism is anchored on the existence of multiple classes and the inheritance kind of relationship.

In a nutshell, inheritance leads to creation of classes which are categorized as either super classes or subclasses. In this case, subclasses inherit attributes and properties of super classes. Polymorphism on the other hand makes use of the methods that perform tasks differently. Hence, the aspect of appearing in “many forms” is signified by the action of performing of a single action in different ways. Polymorphism is perceived to add flexibility to program codes, thus, making them more extensible and maintainable.

Overriding is a form of polymorphism used to dynamically bind a method from a subclass in response to a method call from a subclass object referenced by superclass type. Method overloading is another form of Polymorphism, though some authors perceive it not to be a

form of polymorphism. Method overloading refers to existence of multiple methods with the same name, within the same class. The methods have different method signatures where a call to correct method is resolved at compile time. Overloading is a compile-time activity as opposed to overriding which is a runtime activity. Because of this reason overloading is perceived to be faster than method overriding.

UML Class Diagram Representation for Polymorphism

Considering the fact that polymorphism relies on existence of inheritance, the class diagram in Figure 1 that was used to demonstrate the concept of inheritance is revisited.

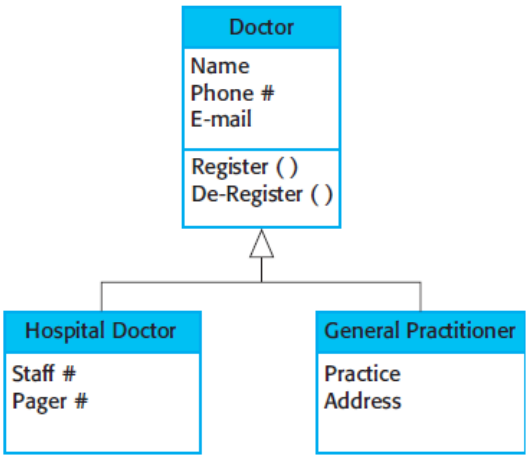

Figure 1. Class Diagram Showing Inheritance [5]

In essence, the attributes and operations associated with the higher-level class, the Doctor class, are also associated with the lower-level classes, Hospital Doctor and Practitioner Doctor classes. Thus, the lower-level class, subclass, inherits the attributes and operations from its superclass. The lower-level class then adds more specific attributes and operations that it possesses.

Considering the model, polymorphism can easily be implemented in case there are methods in the Doctor class that are inherited by the Hospital Doctor and General Practitioner classes. This implementation concept will be demonstrated in the next subsection using a different example.

Implementing Polymorphism through Method Overriding

Implementation of method overriding as a form of polymorphism requires a definition and use of methods with similar names in different classes. The relationship between the classes must be an inheritance. This means that one of the classes must be a parent class while the others are child classes.

For instance, consider a sample case of a Vehicle class that can be inherited by different types or categories of vehicles. These may include a car, minibus, lorry, trailer, among other types of vehicles.

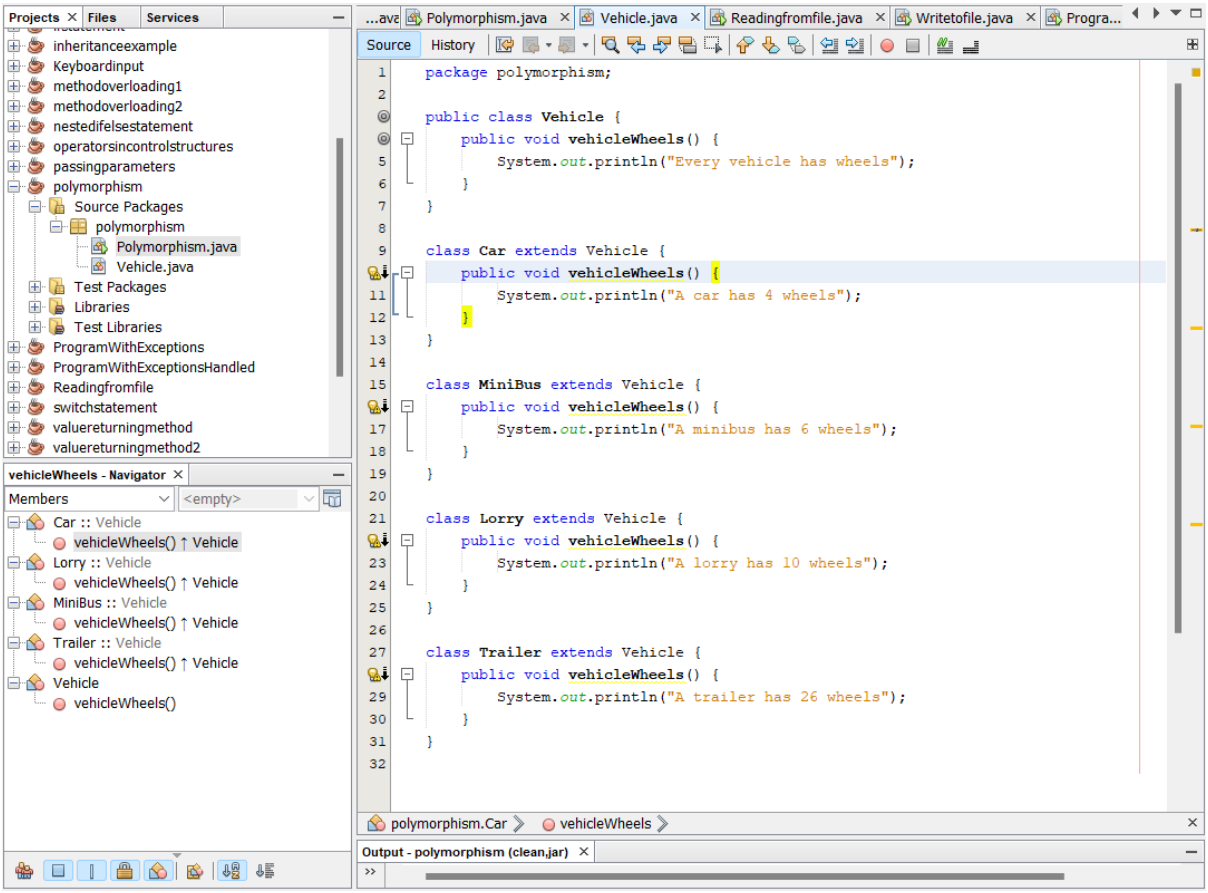

Figure 2. Super and Subclasses

According to Figure 2, each of the vehicle categories has been defined as a class that inherits some properties from the Vehicle, parent class. A method called vehicleWheels() has been defined in the superclass and all other subclasses. The only difference is the fact that each method instance in each of the subclasses has a different implementation. By so doing, the aspect of method overriding is demonstrated.

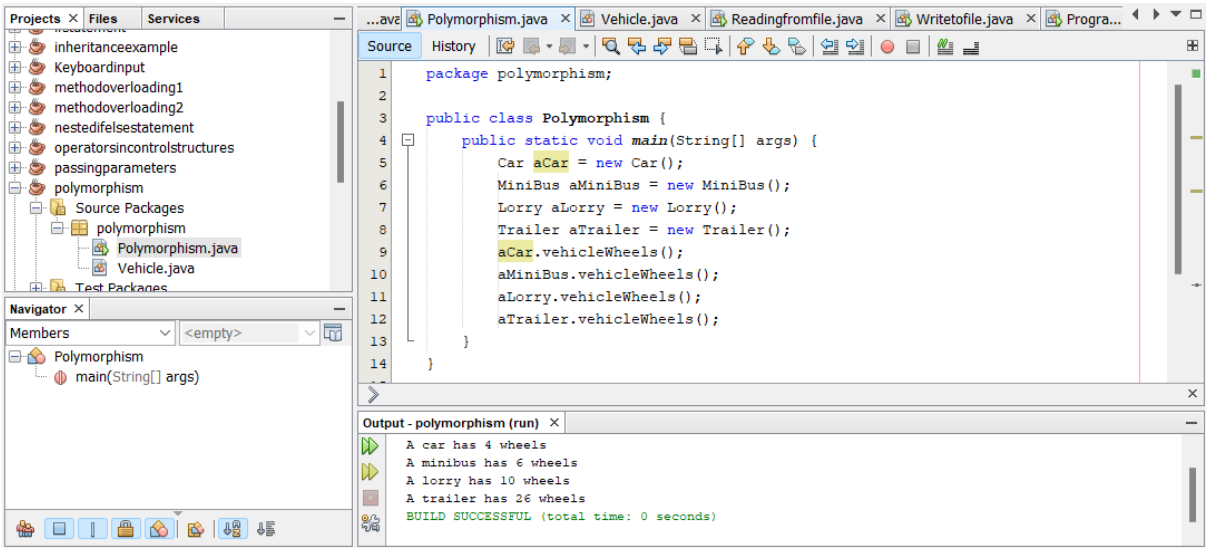

Figure 3. Polymorphism Results

The output shown in Figure 3 shows the vehicleWheels() method in the Car, MiniBus, Lorry, and Trailer subclasses have been executed respectively. All the four methods override the vehicleWheels() method of the Vehicle superclass.

Method Overloading

Method overloading happens in cases where a class has two or more methods with the same method name. furthermore, each of the method should however have a different number and/or type of parameters.

Figure 4 shows an example of method overloading where the Sum class has two sum() methods, each with a different number of parameters. The output clearly shows that the choice of method to execute is automatically determined based on the number of arguments provided at the point of method call. For instance, since the method call in line 18 provided two arguments, the methods defined in lines 5 to 7 executed. On the other hand, since the method call in line 21 had three arguments, the method defined in lines 10 to 12 executed.

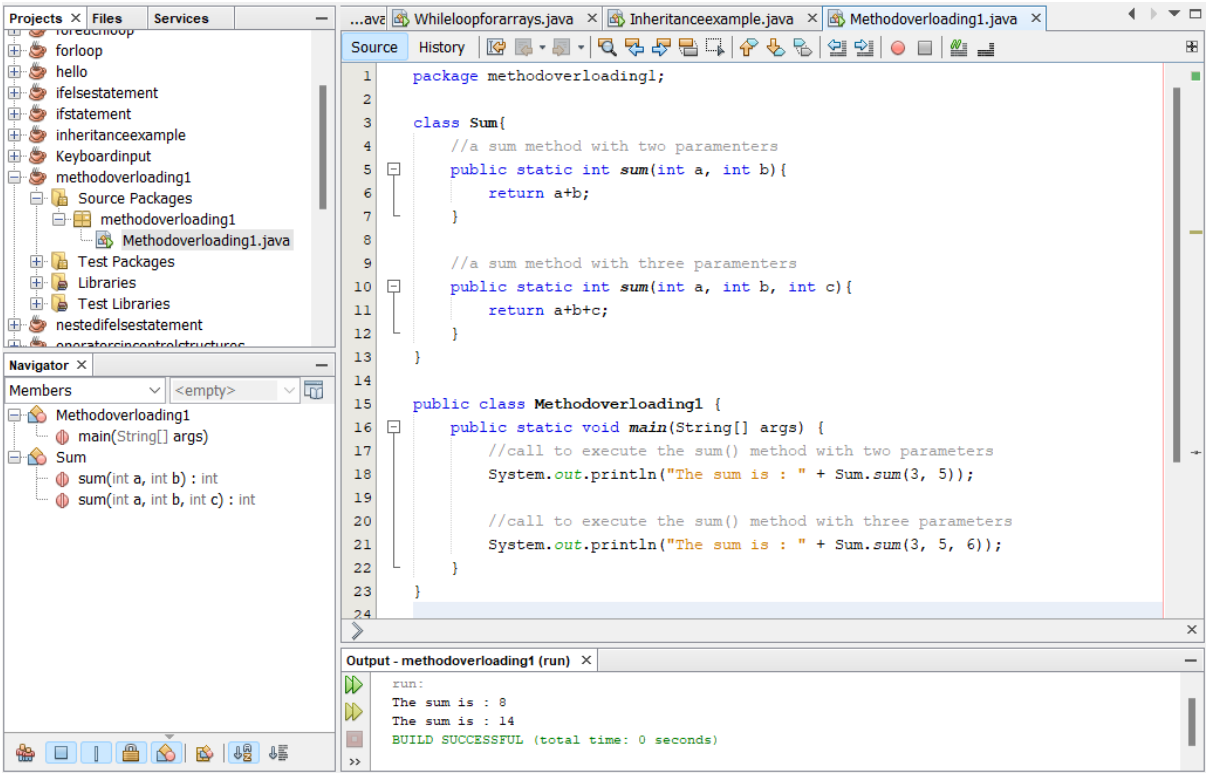

Figure 4. Method Overloading Example 1

Figure 5 shows another example of method overloading. This time round, the Sum class has two sum() methods, with the same number of parameters. However, the parameters are of different data types as shown in lines 5 and 10. The output clearly shows that the choice of method to execute is automatically determined based on the data types for the argument values provided at the point of method call. For instance, since the method call in line 18 provided two integer arguments, the methods defined in lines 5 to 7 executed. On the other hand, since the method call in line 21 provided double and integer arguments respectively, the method defined in lines 10 to 12 executed.

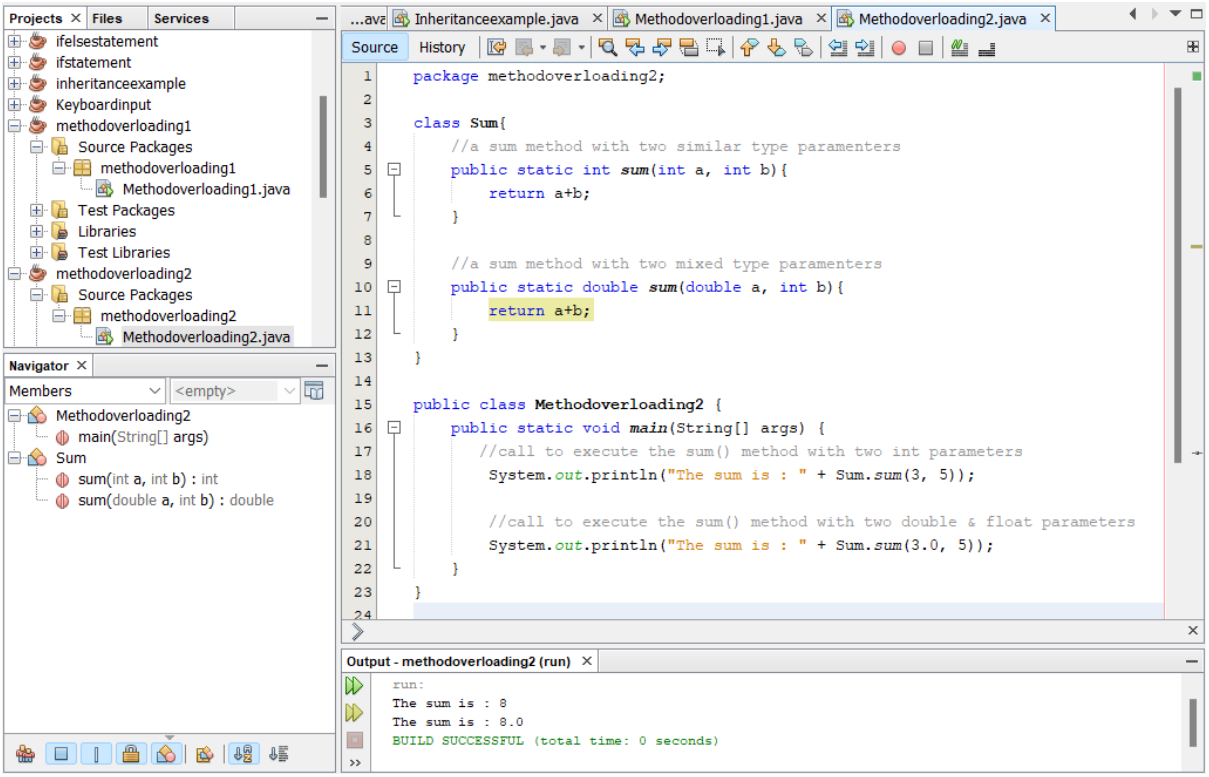

Figure 5. Method Overloading Example 2

The two examples of method overloading depict that fact that overloading can happen based on a variance in number of arguments and/or variance in data types for the parameters. Compilers automatically detect the method to execute by matching the number and type of parameters against arguments.

Differences Between Method Overriding and Overloading

1. Method overloading is used to increase program readability while overriding provides specific implementation of the method that is already provided by its super class.
2. Method overloading is performed within class while method overriding occurs in two classes that have relate through inheritance.
3. Parameters in overloaded methods must be different while in overridden methods must be the same.
4. Overloading occurs at compile time while overriding occurs at run time.

Benefits of Polymorphism

Polymorphism has a number of benefits that include support for code reusability, support for a single variable name for multiple data types, and a reduction in coupling between different functionalities. On the contrary, in realtime, polymorphism ends up raising performance issues, it also reduces code readability. Furthermore, implementation of polymorphism is abit challenging.

Summary

The topic has explored polymorphism as a critical object-oriented concept. Polymorphism refers to appearing in many forms. Method overriding that occurs in classes that embrace inheritance has been discussed and demonstrated using an example. The concept of method overloading that is sometimes disputed to be a type of polymorphism has been equally discussed and demonstrated with examples. Thus, differences between method overriding and overloading have been outlined. Finally, the main benefits of polymorphism has been brought out.

Check Points

1. Describe polymorphism and its benefits in object-oriented programming.
2. Using an example, demonstrate method overriding as a concept of polymorphism.
3. Discuss the relationship between polymorphism and inheritance as applied in object-oriented programming.
4. Using an example, demonstrate any two case scenarios of method overloading as applied in object-oriented programming.
5. Differentiate between method overriding and method overloading as applied in object-oriented programing.

Core Textbooks

1. Joyce Farrell, Java Programming, 7th Edition. Course Technology, Cengage Learning, 2014, ISBN-13 978-1-285-08195-3.
2. Malik, Davender S. JavaTM Programming: From Problem Analysis to Program Design, International Edition, 5th Edition, Cengage Learning.

Other Resources

3. Daniel Liang, Y. "Introduction to Java Programming, Comprehensive." (2011).
4. Malik, Davender S. JavaTM Programming: From Problem Analysis to Program Design, International Edition, 4th Edition, Cengage Learning, 2011.
5. Shelly, Gary B., et al. Java programming: comprehensive concepts and techniques. Cengage Learning, 2012.

References

[1] Farrell, J., Java Programming, 7th Edition. Course Technology, Cengage Learning, 2014, ISBN-13 978-1-285-08195-3.

[2] Malik, D. S., JavaTM Programming: From Problem Analysis to Program Design, International Edition, 5th Edition, Cengage Learning.

[3] Sebester, R. W., Concepts of Programming Languages, 12th Edition, Pearson, 2018, ISBN 0-321-49362-1.