Unit 09: Inheritance

Content Area: **Computer Science**
Course(s):
Time Period: **Marking Period 3**
Length: **4-5 Weeks**
Status: **Published**

Summary

Creating objects, calling methods on the objects created, and being able to define a new data type by creating a class are essential understandings before moving into this unit. One of the strongest advantages of Java is the ability to categorize classes into hierarchies through inheritance. Certain existing classes can be extended to include new behaviors and attributes without altering existing code. These newly created classes are called subclasses. In this unit, students will learn how to recognize common attributes and behaviors that can be used in a superclass and will then create a hierarchy by writing subclasses to extend a superclass. Recognizing and utilizing existing hierarchies will help students create more readable and maintainable programs.

Revision Date: July 2021

CS.9-12.8.1.12.AP.1	Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.2	Create generalized computational solutions using collections instead of repeatedly using simple variables.
CS.9-12.8.1.12.AP.4	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
CS.9-12.8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.DA.2	Describe the trade-offs in how and where data is organized and stored.
CS.9-12.8.2.12.NT.1	Explain how different groups can contribute to the overall design of a product.
CS.9-12.8.2.12.NT.2	Redesign an existing product to improve form or function.
WRK.K-12.P.4	Demonstrate creativity and innovation.
WRK.K-12.P.5	Utilize critical thinking to make sense of problems and persevere in solving them.
TECH.8.1.12.B.CS1	Apply existing knowledge to generate new ideas, products, or processes.
TECH.8.1.12.C.CS4	Contribute to project teams to produce original works or solve problems.
TECH.8.1.12.F.CS2	Plan and manage activities to develop a solution or complete a project.
TECH.8.2.12.E.1	Demonstrate an understanding of the problem-solving capacity of computers in our world.
TECH.8.2.12.E.3	Use a programming language to solve problems or accomplish a task (e.g., robotic functions, website designs, applications, and games).
TECH.8.2.12.E.4	Use appropriate terms in conversation (e.g., troubleshooting, peripherals, diagnostic software, GUI, abstraction, variables, data types and conditional statements).
TECH.8.2.12.E.CS1	Computational thinking and computer programming as tools used in design and engineering.
TECH.9.4.12.CI.1	Demonstrate the ability to reflect, analyze, and use creative skills and ideas (e.g., 1.1.12prof.CR3a).
TECH.9.4.12.CT	Critical Thinking and Problem-solving

TECH.9.4.12.CT.2	Explain the potential benefits of collaborating to enhance critical thinking and problem solving (e.g., 1.3E.12profCR3.a).
TECH.9.4.12.TL.3	Analyze the effectiveness of the process and quality of collaborative environments.

Essential Questions & Essential Understanding

- How does inheritance make programs more versatile?
- How does polymorphism define which methods are run?
- How does the object superclass apply to all subclasses?
- How is the keyword super utilized to import code?
- How might the use of inheritance help in writing a program that simulates real world events?
- What is inheritance of classes?

Objectives

Students Will Know

- how to create a subclass of a class using the constructors, instance variables and methods.
- how to read Unified Modeling Language (UML) to understand relationships between classes.
- how to override methods.

Students Will Be Skilled At

- analyzing similar objects to determine overarching methods and variables that all versions of the object has, in order to create a parent class for all subclasses.
- reusing code from a parent class to limit the additional coding necessary for a subclass.
- understanding relationships between parent classes and their subclasses.

Learning Plan

As a class, building a description for all versions of an item (radio, phone, etc). What do they all remember and what can they all do. Defining instance variables and methods.

Discussion of public vs private aspects of classes.

Create a plan for a classwide Student class. Build together - share code once finished. Class can be of something different, but it gives students a concrete example and flows into next steps easily. Feel free to try something else.

Groupwork developing subclasses for Freshman/Sophomore/Junior/Senior subclass.

Once complete, use a tester class to create objcets of all types, and put them in an array (or ArrayList) to test

all parent methods.

Use this tester class to teach polymorphism - assigning an object of type Senior (subclass) to a reference of type Student (parent)

- At compile time, methods in or inherited by the declared type determine the correctness of a non-static method call
- At run time, the method in the actual object type is executed for a non-static method call

Assessments

Assessments

- Formative: Daily assessments using examples from class notes and CodeHS.com, AP Classroom/Albert Checks for Understanding
- Summative: Teacher-created assessments/projects and CodeHS Computer Science Projects, AP Classroom/Albert Unit Assessments
- Benchmark: Check for understanding benchmark assessments on CodeHS, AP Classroom/Albert/Khan Academy Diagnostics
- Alternative Assessments: Student-centered activities such as a doorbell coding project, game design projects, and other activities involving real world applications

complete performance tasks:

- Students will be able to design subclasses and superclasses using appropriate code.
- Students will be able to write programs using classes in order to model real world objects.

complete quizzes/tests:

- instance variables
- constructors
- accessors & modifiers
- super keyword
- inheritance
- polymorphism - compile time and run time methods

complete sample AP multiple choice questions.

complete sample AP open ended questions.

Materials

District Approved Textbook
Java Concepts for AP Computer Science Study Guide
CollegeBoard AP Classroom Website
CollegeBoard AP Computer Science A Website

Integrated Accomodations & Modifications

Possible accommodations/modification for AP Computer Science A