**Objectives**

**After analyzing the unit topics: Inheritance, you will construct answers to open-ended questions and complete the progress check AP Classroom Unit 9 Personal Progress Check, MCQ.**

**What do I do?**

1. **Read all instructions before beginning.**
2. **Open the College Board AP Classroom, Unit 9 Inheritance.**
3. **Watch the specified videos. Answer the following questions in complete sentences in your Word document.**

**AP Classroom Unit 9 Inheritance**

**Watch 9.1 Videos 1 & 2 Creating Super Classes and SubClasses (Is-A)**

1. **Why use inheritance? (6 points) Inheritance is used in object-oriented programming to promote code reusability and to establish hierarchical relationships between classes. By using inheritance, a subclass can inherit the attributes and methods of its superclass, allowing for the creation of more specialized classes based on existing ones. This reduces code duplication and facilitates the organization of related classes into a logical hierarchy.**
2. **How many super classes can a subclass have? (4 points) In Java, a subclass can have only one direct superclass. This is because Java supports single inheritance, meaning that a class can extend only one other class. However, a class can indirectly inherit from multiple classes through a chain of inheritance.**
3. **If Hamburger is a Sandwich and Sandwich is a LunchItem, can Hamburger extend LunchItem? (4 points) No, Hamburger cannot directly extend LunchItem.**
4. **Why? (4 points) In the given scenario, Hamburger is already a subclass of Sandwich, and Sandwich is a subclass of LunchItem. Java does not support multiple inheritance, so a class cannot directly extend more than one class. Since Hamburger already extends Sandwich, it cannot also directly extend LunchItem. However, Hamburger will inherit the attributes and methods of LunchItem indirectly through Sandwich.**
5. **Can Hamburger access the attributes and methods of both Sandwich and LunchItem? (4 points) Yes, Hamburger can access the attributes and methods of both Sandwich and LunchItem.**
6. **Why? (4 points) Since Hamburger is a subclass of Sandwich, and Sandwich is a subclass of LunchItem, Hamburger inherits all the public and protected attributes and methods from both Sandwich and LunchItem through the chain of inheritance. This means that an instance of Hamburger can access and use the attributes and methods defined in its direct superclass (Sandwich) and its indirect superclass (LunchItem).**

**Watch 9.2 Video 2 Writing Constructors for Subclasses**

1. **Given: Hamburger is a Sandwich and Sandwich is a LunchItem. If Hamburger has a Constructor that does not specifically call the Constructor above it, when its Constructor gets called, will Constructors in the hierarchy above get called? (3 points) Yes, even if the Hamburger constructor does not explicitly call the constructor of its superclass (Sandwich), the constructors in the hierarchy above it will still be called.**
2. **If so, what Constructors will be called? (3 points) When an instance of Hamburger is created, the constructors will be called in the following order:**
   1. **LunchItem constructor (if it exists)**
   2. **Sandwich constructor**
   3. **Hamburger constructor**
3. **If your Constructor wants to call the super Constructor, where must the call be placed: (3 points)**
   1. **In the first line of the Constructor**
4. **What is the sole purpose of Contructors? (3 points) The sole purpose of constructors is to initialize the object's state (attributes) when an instance of the class is created. Constructors are called during object creation to set the initial values of the object's attributes and perform any necessary setup tasks.**
5. **Are Constructors inherited? (3 points) No, constructors are not inherited by subclasses. Each class must define its own constructors. However, a subclass constructor can call its superclass constructor using the super keyword.**
6. **Can a class have more than one Constructor? (3 points) Yes, a class can have more than one constructor. This is known as constructor overloading. Multiple constructors can be defined with different parameter lists to allow for different ways of initializing an object.**
7. **If a Subclass has a method with the same signature as the Superclass, which method is called when a Subclass object, subC, executes subC.method()? (5 points) 1. The Subclass version**
8. **If a Subclass has a method with the same name, but a different signature, did it override the Superclass method with the same name? (5 points) No, if a subclass has a method with the same name but a different signature (different parameter list) than the superclass method, it does not override the superclass method. Instead, this is called method overloading. Overriding requires the method signature (name and parameter list) to be exactly the same in the subclass as in the superclass.**
9. **What is the difference between Overriding and Overloading methods? (5 points) Overriding occurs when a subclass defines a method with the same signature (name and parameter list) as a method in its superclass. The overriding method in the subclass provides a different implementation or behavior than the method in the superclass. When an overridden method is called on an instance of the subclass, the subclass's implementation is executed. Overloading, on the other hand, occurs when a class has multiple methods with the same name but different parameter lists. The methods can have different numbers, types, or orders of parameters. Overloaded methods provide different ways to invoke the method based on the arguments passed. The compiler determines which overloaded method to call based on the arguments provided during the method invocation.**
10. **If your Subclass method wants to call the Superclass method, where must the call be placed: (5 points) 1. Anywhere in the Subclass method**
11. **If a Subclass calls super, does it apply to (8 points) The class it extends**
12. **If Class1 Is-A Class2, which of the following is allowed: (8 points) Class2 c2 = new Class1();**
13. **If a Parent and Child both have a methodOne(), given: Parent p = new Child(); Which version of methodOne() will be called when p.methodOne() executes? (5 points) The Child's version of methodOne()**
14. **Given the information in the previous question and Child c = newChild(); If we have a method in a different class defined as follows: newMethod(Parent oldPerson) Specify all of the following method calls that are legal: (5 points)**
    1. **\* newMethod(p)**
    2. **\* newMethod(c)**
    3. **\* newMethod((Parent)c)**
15. **What class is at the top of Java's Inheritance hierarchy? (5 points) The class at the top of Java's inheritance hierarchy is the Object class. Every class in Java, whether explicitly specified or not, inherits directly or indirectly from the Object class.**
16. **What two methods are included in this Java super class? (5 points) The Object class includes several methods, but two commonly used methods are:**
    1. **1. toString(): Returns a string representation of the object. This method is often overridden by subclasses to provide a meaningful string representation of the object's state.**
    2. **2. equals(Object obj): Compares the object with another object for equality. By default, this method checks if two object references point to the same object in memory. It is often overridden to provide a custom equality comparison based on the object's attributes.**