**The is the change.**

**119. Generalization and Specialization.**

1. Generalization: The derivation is bottom up.

2. Specialization: The derivation is top-bottom.

3. We can see both in the form of hierarchy.

4. Specialization is achieved using inheritance.

5. Generalization is achieved using interface.

6. If you want to achieve something between Specialization or Generalization the it can be achieved it using Abstract Class.

**120. What is Inheritance?**

1. Inheritance is the process of acquiring of features of existing class into new class.

2. **“extends”:** It is used for the purpose of inheritance. It means extension.

3. Super-sub is most commonly used term is used in java.

4. Inheriting the class makes everything available but not accessible.

**121. Practicing the inheritance.**

**122. Student Challenge: Account, Customer, etc.**

Bank Account:

1. class Account:

a. Saving Account:

b. Loan Account:

2. Common properties:

a. accno.

b. name.

c. address.

d. phone number.

e. date of birth.

f. balance.

3. Common Methods

a. get methods.

b. set methods.

c. constructors.

4. Saving Accounts:

Methods:

a. Deposit.

b. Withdraw.

c. Fixed Deposit.

e. liquidate.

5. Loan Accounts:

a. pay emi.

b. top up loan.

c. repayment.

**123. Constructors in Inheritance.**

1. When we create of child class the parent class constructor is executed first and then the child class constructor. When create object of child class .

2. The constructors are executed from top to bottom.

**124. Parameterized Constructor.**

1. **“super”** keyword refers to super class.

2. super(param..) means it is calling the constructor. The super key word should be used in the very first line.

3. We can’t access the parent class using the super in the grand child class. It is used to access the direct super class.

**125. this vs super**

1. **“this”** is a reference to current or present object.

2. Use of this is when there is any name conflict between parameters and the properties.

3. **“super”** is a reference to super class.(It is reference to the set of properties coming from super class.)

4. Use of super is to call the parameterized constructor on super class or access members of super class.

**126. Method Overriding.**

1. **Method Overriding:** It is redefining the method of super class into child class.

Method of super class is shadowed and child class method is visible.

2. **Dynamic Method Dispatch:** A super class reference holding the object of subclass and overridden method is called. The method of an object is called. But we can’t do opposite.

3. Method will called depending on the object not on the reference.

4. **Philosophy:**

**127. Practicing: Method Overriding.**

**128. Example #1: Method Overriding.**

1. We can call smart tv a tv but we can’t called tv a smart tv.

2.

**129. Example #2: Method Overriding.**

**130. Dynamic Method Dispatch.**

1. **Dynamic Method Dispatch** is useful to achieving runtime polymorphism using method overriding.

2. You can have supe class reference and object of sub class. Other way it is not allowed.

3. We can only call inherited and overridden methods.

4. A Super class holding object of sub class can call only methods of super class (which are inherited and overridden).

5. This is runtime polymorphism.

6. Methods are called based on object not based on the reference.

**131. Practicing Dynamic Method Dispatch.**

**132. Do’s and Don’ts of Overriding.**

1. Signature of the method must be same. Otherwise it will become method overloading or totally different method.

2. We can return object using method.

3. We cannot override static and final methods.

4. There are three access modifiers.

a. public:

b. protected:

c. public:

**133. Polymorphism using overloading and overriding.**

1. **Polymorphism:** Same name, different actions.

2. **Method Overloading:** This is a compile time polymorphism. It’s achieved in same class and child class.

3. **Method Overriding:** This is a runtime polymorphism. It achieved. It’s achieved only in inheritance.

Compiler cannot decide which method to be called as object is decided at runtime. Even though compiler compiles the code the decision is made in runtime.