OOD

1. Why you interested in Convertus

Convertus is one of the largest automotive digital marketing agencies in Canada. It focuses on a promising area which means the company will continue to grow in the next 5 years. As a fast-growing company, software engineers will have the opportunity to face many challenging and interesting problems and learn many new technologies. These attract me a lot.

OOP Design

面向对象: **object-oriented**
面向过程: **procedure-oriented**

BASIS FOR COMPARISON	**POP**	**OOP**
Approach	Top-down	Bottom-up
Basis	Main focus is on "how to get the task done" i.e. on the procedure or structure of a program .	Main focus is on 'data security'. Hence, only objects are permitted to access the entities of a class.
Division	Large program is divided into units called **functions.**	Entire program is divided into **objects.**
Inheritance	No inheritance	public, private, protected, friend
Data sharing	Global data sharing	Data is shared among the objects through the member functions.

1. Four basic principles of Object Oriented Programming

1. Encapsulation(封装) [ɪn,kæpsə'leʃən]

Encapsulation is the mechanism of **hiding of data implementation** by **restricting access to public methods**.

```
public class Employee {
    private String name;
    private Date dob;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Date getDob() {
        return dob;
    }
    public void setDob(Date dob) {
        this.dob = dob;
    }
}
```

2. Abstraction(抽象)

Its main goal is to **handle complexity** by **hiding unnecessary details** from the user.

3. Inheritance(继承) [ɪn'hɛrɪtəns]

Inheritances expresses "is-a" and/or "has-a" relationship between two objects. In Java, concept of **"is-a"** is based on **class inheritance (using extends)** or **interface implementation (using implements)**.

4. Polymorphism(多态) [ˌpɑlɪ'mɔrfɪzm]

Polymorphism means **"many forms"**, and it occurs when we have many classes that are related to each other by inheritance.

For example, think of a **superclass** called **Animal** that has a method called **animalSound()**. Subclasses of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own **implementation of an animal sound (the pig oinks, and the cat meows, etc.)**:

2. Override and Overload

Method Overloading	MethodOverriding
1. It occurs with in the same class. 2. Inheritance is not involved. 3.One method does not hide another. 4.Parameters must be different. 5.return type may or may not be same. 6. Access modifier & Non access modifier can also be changed.	1. It occurs between two classes i.e., Super class and a subclass. 2. Inheritance is involved. 3. child method hides that of the parent class method. 4. Parameters must be same. 5. return type must be same. 6. Access modifier should be same or increases the scope of the access modifier. Non access modifier – • **final :** if a method can contain final keyword in a parent class we cannot override. • **static:** if a method can contain static keyword child cannot override parent class methods but hide (child).

Overriding

```
class Dog{
    public void bark(){
        System.out.println("woof ");
    }
}
class Hound extends Dog{
    public void sniff(){
        System.out.println("sniff ");
    }

    public void bark(){
        System.out.println("bowl");
    }
}
```

Same Method Name, Same parameter

Overloading

```
class Dog{
    public void bark(){
        System.out.println("woof ");
    }

    //overloading method
    public void bark(int num){
        for(int i=0; i<num; i++)
            System.out.println("woof ");
    }
}
```

Same Method Name, Different Parameter