```java
public class C1 {
	C1(int x){
		System.out.println("Construtor C1");
	}

	void imprimir(){
		System.out.println("Imprimir C1");
	}
}


public class C2 extends C1 {
	C2(){
		super(0);
		// super(1); NÃO FUNCIONA: O CONSTRUTOR DA SUPER CLASSE PRECISA SER O PRIMEIRO COMANDO NO CONSTRUTOR DA SUB-CLASSE
		System.out.println("Construtor C2");
	}

	void imprimir(){
		System.out.println("Imprimir C2");
	}
}


public interface I1 {
	int constanteI1 = 34;

	public void metodoInterfaceI1();

	void imprimir();

}

public class C3 extends C2 implements I1{
	// C3 não precisa ter construtor pois o construtor de C2 não tem parâmetros
	// então será chamado automaticamente pelo construtor padrão de C2

	// Note que o método imprimir existe em C2 e em I1
	// Por existir em I1 é necessário reimplementá-lo em C3 (não pode simplesmente herdá-lo)
	public void imprimir(){
		System.out.println("Imprimir C3");
	}

	void imprimirAlternativo(){
		super.imprimir();
		// super.super.imprimir();  NÃO FUNCIONA não podemos usar "super.super"
	}

	void imprimirAlternativo2(){
		((C1)this).imprimir(); // vai executar o imprimir de C3 (e não de C1) é a mesma coisa de "imprimir();"
	}

	// as implementações de métodos de interface precisam ser públicas
	public void metodoInterfaceI1() {
		System.out.println("Imprimindo valor da constante de I1: " + I1.constanteI1);
	}

}
```

```java
public class C4_Exec {

	public static void main(String[] args) {
		C3 x = new C3();
		x.imprimir();
		System.out.println("Alternativo:");
		x.imprimirAlternativo();
		System.out.println("Alternativo2:");
		x.imprimirAlternativo2();
	}
}

/* RESULTADO DA EXECUÇÃO
Construtor C1
Construtor C2
Imprimir C3
Alternativo:
Imprimir C2
Alternativo2:
Imprimir C3
*/
```