Programação Orientada a Objetos
Lista de Exercícios 1

1. Visibilidade de Métodos em Herança

   a) Responda as questões e apresente o resultado conforme os comentários do código abaixo. b) Para cada quadro, faça uma crítica comentando o que o mesmo está fazendo e possíveis problemas. Resolva o problema sem executá-lo em computador. Depois confira os resultados executando o mesmo.

```
public class Teste {
	public static void main(String[] args){


	A a1,a2;
	B b1,b2;
	a1 = new A();
	a2 = new B(); // //  lowcast ou downcast? Qual a consequência?
```

```
	a1.setX(10);
	a2.setX(20);
	a1.ma(); // resultado:
	a2.mb(); // resultado:
```

```
	((B) a2).mb(); // resultado:
	b1 = new B();
	b1.setX(10);
	b1.ma();  // resultado:
```

```
	try {
	   b2 = (B) new A(); //  lowcast ou downcast? Qual a consequência?
	}
	catch (Exception e){
		e.printStackTrace();
	}
	finally {

	   b2 = new B();
	   ((A) b2).setX(20); // resultado:
	   b2.mb(); // resultado:
	   b2.ma(); // resultado:
	}
  }

}
class A {
	private int x;
	public void ma(){
		System.out.println(getX());

	}
		public int getX() {
			return x;
		}
		public void setX(int x) {
			this.x = x;
```

```
            }

}
class B extends A {
       private int x;
       public void mb() {
         System.out.println(getX());
       }

       public int getX() {
             return x+1;
       }
       public void setX(int x) {
             this.x = x+1;
       }

}
```

2. Escreva os nomes dos métodos que serão chamados pelo código dado na tabela abaixo Utilize Classe::metodo para identificar o método (exemplo: A::m para método m da classe A). O código está em Java e portanto lembre-se de que todas as variáveis são de fato ponteiros. É possível que exista um erro de tipos e que alguma chamada de métodos entre em loop infinito. Se isto acontecer, aponte o fato e continue a escrever os métodos que seriam chamados se não houvesse um erro ou loop innito.

A a = new C(); a.f();	
a.s();	
B b = new B(); b.f();	
a = new A(); a.f();	

```
interface Printable {
  void print();
}
class A implements Printable {
  void f() { g(); }
  void g() { f(); }
  void r() { }
  public void print() { System.out.println("A::print"); }
}
class B extends A {
   void r() { this.f(); }
   void g() { }
   void s() { }
}
class C extends B {
  void f() { super.g(); }
  void g() { this.print(); }
  void s() { super.s(); }
}
```