Lista de Exercícios 10
Programação Orientada a Objetos

Exercício 01) Explique os principais conceitos da Programação Orientada a Objetos: encapsulamento, herança, composição e polimorfismo. Dê um exemplo de código para cada um deles, diferentes dos vistos nos slides, em qualquer linguagem de programação.

Exercício 02) Considere o seguinte trecho de código em Java. Mostre o que será impresso e explique quando é feita a associação do tipo do objeto para chamar o método (chamada polimórfica).

```
01: class Pessoa {
02:   private String nome;
03:   public Pessoa(String nome) {
04:     this.nome = nome;
05:   }
06:   public void print() {
07:     System.out.println(
08:       "Nome: " + nome);
09:   }
10: }
11: class PessoaFisica extends Pessoa {
12:   private long cpf;
13:   public PessoaFisica(
14:         String nome, long cpf) {
15:     super(nome);
16:     this.cpf = cpf;
17: }
```

```
18:   public void print() {
19:     super.print();
20:     System.out.println(
21:       "CPF: " + cpf);
22:   }
23: }
24: class Program {
25:   public static void main(
26:         String args[]) {
27:     Pessoa p = new
28:     PessoaFisica("Joao",
29:         12345678901L);
30:     p.print();
31:   }
32: }
```

Exercício 03) Considere o programa do exercício 2:

a) Reescreva-o em C++ simulando o mesmo comportamento de Java.
b) É possível modificar o código de forma que a chamada `print` chame o método da superclasse ao invés da subclasse? Se sim, mostre como.

Exercício 04) Padrões de projeto documentam soluções para diversos problemas recorrentes em programação. Um padrão de projeto bastante conhecido é o *Singleton*. Basicamente, este padrão descreve como declarar uma classe que possua, no máximo, uma instância. Mostre então a declaração de uma classe `S` em C++ que seja um *Singleton*. Esta classe deverá ter um método estático `instance()` que sempre será usado para se criar objetos do tipo `S`. Caso ainda não existam instâncias de `S`, o método instancia um objeto dessa classe. Caso já exista uma instância, o método deverá retornar essa instância.

Exercício 05) Java implementa o polimorfismo universal paramétrico como *Generics*, enquanto C++ o implementa como *templates*. Os dois modelos servem para o mesmo propósito, mas são fundamentalmente diferentes em sua implementação. Discuta sobre como essas duas linguagens de programação implementam esse tipo de polimorfismo.

Exercício 06) O trecho de programa a seguir em Java apresenta algum erro? Se sim, descreva-o e indique o que pode ser feito para corrigi-lo. Caso contrário, mostre o que será impresso.

```
01: class E {
02:   public static int x = 0;
03:   public static void p() {
04:     x++;
05:     System.out.println("E::p");
06:   }
07:   public void q() {
08:     System.out.println(x);
09:   }
10: }
```

```
11: class Program {
12:   public static void main(String[] args) {
13:     System.out.println(E.x);
14:     E.p();
15:     System.out.println(E.x);
16:     E e = new E();
17:     System.out.println(e.x);
18:     e.q();
19:   }
20: }
```

Exercício 07) Mostre o que será impresso pelos seguintes programas em C++:

a) Prog1.cpp

```
01: class A {
02:   public:
03:     virtual void f() { cout << "A::f()\n"; }
04:     void g() { cout << "A::g()\n"; }
05: };
06: class B: public A {
07:   public:
08:     void f() { cout << "B::f()\n"; }
09:     void g() { cout << "B::g()\n"; }
10: };
```

```
11: int main() {
12:   B b;
13:   A *ap = &b;
14:   B *bp = &b;
15:   ap->f();
16:   ap->g();
17:   bp->f();
18:   bp->g();
19: }
```

b) Prog2.cpp

```
01: class A {
02:   public:
03:     int x;
04: };
05: class B1: public A {
06:   public:
07:     void f() {
08:       cout << x << endl;
09:     }
10: };
11: class B2: public A {
12:   public:
13:     void f() {
14:       cout << x << endl;
15:     }
16: };
```

```
17: class C: public B1, public B2 {
18:   public:
19:     void g() {
20:       B1::x = 4;
21:       B2::x = 5;
22:       B1::f();
23:       B2::f();
24:     }
25: };
26: int main() {
27:   C c;
28:   c.g();
29: }
```

Exercício 08) Reimplemente em C++ o programa para cálculo da área de figuras geométricas visto em aula (o código em Java pode ser encontrado nos slides).

Exercício 09) Reimplemente em Java a estrutura de dados Pilha vista em aula (o código em C++ pode ser encontrado nos slides). Você deve usar como estrutura de dados um arranjo para armazenar os elementos. Você não pode utilizar estruturas de dados de alto nível, como `List` e `Vector`.