1) Em programação e na orientação a objetos, uma classe é um Tipo abstrato de Dados (TAD); ou seja, uma descrição que abstrai um conjunto de objetos com características similares (um projeto do objeto), é um código da linguagem de programação orientada a objetos que define e implementa um novo tipo de objeto, que terão características (atributos) que guardaram valores e, também funções específicas para manipular estes.

2) Um objeto é uma forma computacional de representar algo da área de interesse do problema, podendo esse algo ser abstrato ou concreto. Objetos podem se vincular a programação hiperativa, já que em sua maioria a programação orientada a objetos segue passo a passo o que deve ser feito com o objeto, consequentemente entrando nesse espectro da programação.

3) A Herança possibilita que as classes compartilhem seus atributos, métodos e outros membros da classe entre si. Para a ligação entre as classes, a herança adota um relacionamento esquematizado hierarquicamente. Na Herança temos dois tipos principais de classe:

- Classe Base: A classe que concede as características a uma outra classe.
- Classe Derivada: A classe que herda as características da classe base.

Dentro disso existem a herança pública e a privada, que possuem diferenças entre si, a pública permite acesso livre aos membros da classe base, enquanto na privada as informações estão presentes, mas só podem ser acessadas através de funções públicas ou protegidas da classe base.

4) O uso de métodos virtuais permite tratar diferentes classes com um mesmo pedaço de código, desde que sejam derivadas de uma mesma classe base. Tais conceitos também descrevem o polimorfismo, já que possuem o mesmo intuito, sando assim o métodos virtuais são uma forma de polimorfismo na programação.

5) Os tipos de encapsulamento têm como objetivo delimitar cada classe e manter cada uma realizando suas operações sem interferência externa, além de poderem impedir acessos indevidos dentro do código.

6) Construtores são basicamente funções de inicialização de uma classe, as quais são invocadas no momento em que objetos desta classe são criadas. Eles permitem inicializar campos internos da classe e alocar recursos que um objeto da classe possa demandar, tais como memória, arquivos, semáforos, soquetes, etc.

Destrutores realizam a função inversa: são funções invocadas quando um objeto está para ser “desativado”. Caso um objeto tenha recursos alocados, destrutores devem liberar tais recursos. Por exemplo, se o construtor de uma classe alocou uma variável dinamicamente com new, o destrutor correspondente deve liberar o espaço ocupado por esta variável com o operador delete.

7) Pode ser utilizar na implementação de uma classe Pessoa que contenha um método de validar CPF:

static bool isCPFValido(string);

Desta forma, podemos chamá-la a partir de código externo à classe sem a necessidade de criar uma nova instância da mesma.

8)

9) As classes e objetos criados por herança são fortemente acoplados porque alterar a classe base ou classe pai em um relacionamento de herança corre o risco de quebrar seu código.

As classes e objetos criados por meio da composição são fracamente acoplados, o que significa que você pode alterar mais facilmente as partes do componente sem quebrar seu código.

Em geral usar composição traz mais vantagens do que utilizar herança. Os projetos tendem a ser mais simples e reutilizáveis em se favorecendo a composição ao invés da herança.

10) C2-f1

C1-f1

C2-f2

C2-f1