**1. Defina o que é feito na etapa de *Análise* e o que é feito na etapa de *Projeto* ao desenvolver software.**

*Na etapa da análise é conversado com o cliente para analisar o que ele quer e fazer o levantamento de requisitos para o sistema, especificar o que será usado, especificar os requisitos, definir tudo sobre o software antes de colocar a mão na massa. Na parte do Projeto é onde se coloca em ação o que o cliente pediu sobre o software, fazer com que o que o cliente pediu na análise seja entregue ao mesmo.*

**2. Enumere as vantagens da abordagem Orientada a Objetos para o desenvolvimento de software.**

*Com a programação orientada a objeto a programação e manutenção fica mais fácil de entender, de produzir. A programação orientada a objeto ajuda na organização e resolve muitos problemas enfrentados pela programação procedural. O código fica menor, mais limpo, mais rápido.*

**3. Qual é o motivo de levantar Requisitos Funcionais para desenvolver software e o que faz parte de uma descrição de Requisitos Funcionais?**

*O motivo é porque o cliente quer ver a coisa acontecer com facilidade que é o que os requisitos funcionais fazem, é o que o cliente vai ver, como a imagem do software, um botão, uma tela com alguma informação, então tudo isso é o que o cliente quer ver e isso é o motivo de levantar esses requisitos funcionais.*

**4. Mostre como instanciar um objeto da classe ContaBancária em Java fornecendo o CPF (um string) do titular como argumento. Com o objeto resultante, faça um depósito de R$100,00 e imprima o saldo. Você pode escolher nomes apropriados para os métodos.**

*Ta no GitHub*

**5. Explique o que é um Iterator em Java. Qual é sua principal vantagem?**

*Iterator é um padrão para percorrer listas, conjuntos, mapas etc.É o que pode ser chamado de "cursor" se estivesse escrevendo stored procedures em algum banco SQL.*

**6. Mostre a implementação de uma classe ContaBancária. Invente atributos e métodos.**

*Ta no GitHub*

**7. Explique a diferença de funcionamento entre um "return" e um "throw". Seja específico.**

*Quando usamos o return ele irá nos retornar algo. Quando usamos o é para tratar algum erro que daria uma exceção se não fosse tratado.*

**8. Mostre, usando Java, como especializar uma classe ContaBancária para criar uma ContaCorrente e uma ContaPoupança**.

**9. Explique as vantagens e desvantagens do polimorfismo. Dê exemplos.**

*Uma das inúmeras vantagens de se usar polimorfismo é que ele é um mecanismo de herança que permite a criação de classes a partir de outras já existentes desde que exista a relação “é um” entre as classes criadas a partir da subclasse. Também permite o aumento do software de forma mais controlada. A desvantagem­ é que se não usar o polimorfismo corretamente acarretara problemas quando for instanciar objetos que não foram criados corretamente.*

**11. Qual é a diferença entre "herança de tipo" e "herança de implementação.”?**

*Quando usamos a herança de tipo a classe herda todos os atributos públicos da classe pai. Já quando usamos a herança de implementação é possível herdar o mesmo método e a mesma assinatura em várias interfaces.*

**12. Quais são as vantagens e desvantagens de acoplamento forte entre objetos?**

*Classes com acoplamento forte depende muito (em geral sem necessidade) de outras classes. Isso pode trazer problemas tais como:*

*- Classes difíceis de aproveitar tendo em vista que sempre que esta for utilizada todas as outras das quais ela depende devem estar presentes;*

*- Alterações nas classes relacionadas podem forçar mudanças locais e são difíceis de compreender isoladamente.*

**14. O que é uma "responsabilidade de uma classe"? Por que queremos minimizar o número de responsabilidades? "Mais" não seria melhor?**

*É o que é de responsabilidade da classe fazer, a competência da mesma. Uma classe deve fazer apenas uma coisa, deve fazê-la bem e deve fazer somente ela. Se uma classe tem mais de um motivo para ser alterada, ela não segue este princípio. Queremos minimizar o numero de responsabilidades para aplicar o princípio da responsabilidade única, que é importante para uma arquitetura madura e sustentável. Quando começamos a dar valor a princípios como este, vemos que estamos amadurecendo e fazendo melhor o que gostamos de fazer: software.*