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

Analisar os requisitos levantados e projetar todo o software a partir deles (escolher tecnologias e serviços a serem utilizados – Prova de Conceito).

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

Reutilização, manutenção, custo e tempo, nível de abstração, padronização, segurança, dentre outras vantagens.

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

Requisitos funcionais definem o comportamento do sistema. Exemplo: Cadastro usuário.

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.

ContaBancaria contaBancaria1 = new ContaBancaria(“123123”);
contaBancaria1.setDeposito(1000.00);
**System**.**out**.**println**(getSaldo);

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

É um padrão de projeto. Esse padrão **Iterator** permite acessarmos um a um os elementos de um agregado mesmo sem saber como eles estão sendo representados, assim torna-se irrelevante se a coleção de objetos está num ArrayList, HashTable ou que quer que seja.

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

public class ContaBancaria {

private int id;

private String senha;

private double saldo;

private double saque;

```
public ContaBancaria(int id, String senha, double saldo, double saque)
{
super();
this.id = id;
this.senha = senha;
this.saldo = saldo;
this.saque = saque;
}
public int getId() {
return id;
}

public void setId(int id) {
this.id = id;
}

public String getSenha() {
return senha;
}
public void setSenha(String senha) {
this.senha = senha;
}

public double getSaldo() {
return saldo;
}

public void setSaldo(double saldo) {
this.saldo = saldo;
}
```

```
public double getSaque() {
return saque;
}

public void setSaque(double saque) {
this.saque = saque;
}
}
```

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

Throw - Para a execução do método e lança uma exceção

Return - Retornar um valor.

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.

Com polimorfismo podemos um maior controle sobre as subclasses geradas a partir de uma classe pai. Podemos dar vários comportamentos ao mesmo método.

10. Explique a afirmação: "Em Java, o conceito de interfaces permite obter mais polimorfismo do que seria possível com classes abstratas".

Nas classes abstratas pelo fato de poder ter métodos implementados, eles não poderão ser modificados. Já na interface, as assinaturas de métodos serão implementadas pela classe que compôs da maneira que a classe quiser.

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

Herança de classe (ou de implementação), define a implementação de um objeto em função da implementação de outro e Mecanismo para compartilhamento de código.
Herança de tipo (ou de interface), define quando um objeto pode ser utilizado no lugar do outro ou cumprindo a mesma promessa que o outro prometeu.

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

A desvantagem é a dependência de um método para o funcionamento dos demais métodos, por exemplo. A alteração desse método tem grande impacto sobre os demais.

Vantagem do forte acoplamento é que pode deixar os objetos mais seguros e difíceis de serem modificados, aumentando a coesão.

13. Ao falar de boa programação, fala-se: "A decomposição deve esconder algo." O que poderia ser escondido, por exemplo?

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

Responsabilidade de uma classe é o comportamento que ela possui. Cada classe deve ter sua responsabilidade específica.

15. Por que modelar papeis (roles) através de herança é inferior a modelá-los através de composição?

Porque através de composição eu estou acoplando menos o meu projeto e a classe que implementar poderá fazer os métodos da maneira que ele quiser.