Exercícios MAP – Rodolfo Almeida

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

A análise é um aspecto importante no Gerenciamento de Projetos, é a responsável por coletar dados indispensáveis, necessários, exigências de que o usuário necessite para solucionar um problema e alcançar seus objetivos. Assim como determinar as suas expectativas de um usuário para determinado produto.

Projeto é a representação significativa de alguma coisa que será construída. Em engenharia de software, o Projeto de Software é a fase de desenvolvimento, na qual são feitos modelos com todas as entidades que serão construídas posteriormente a partir dos requisitos do sistema.

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

- A codificação fica mais próxima do cenário real do problema a ser resolvido: é muito mais simples ter uma classe pessoas que faz o cadastro, atualização, exclusão e busca do que uma função para cada ação necessária.

- A manutenção futura fica mais simples e rápida: você somente altera as classes que são necessárias, sem precisar ficar procurando funções perdidas em meio aos arquivos para alterar algo.

- Maior reutilização de código: imagine que você crie uma classe que faça as operações de INSERT, UPDATE e DELETE do banco de dados, e crie uma classe específica para SELECTs, durante todo o restante do sistema você vai estar usando as mesmas classes, pois se bem planejadas eles se tornam muito mais dinâmicas do que uma simples função.

- Padronização do sistema: criando um sistema em programação procedural, ele fica com o padrão do programador ou da empresa, já em OOP ele fica com o padrão da OOP, ou seja, o sistema será "entendível" para qualquer desenvolvedor futuramente.

- Segurança: é muito mais difícil burlar um sistema orientado a objetos do que um sistema procedural, uma vez que não basta simplesmente inserir alguma função ou algo do gênero, pois é preciso instanciar objetos para que a "coisa funcione". Ainda existe a possibilidade de definir propriedades e métodos publicos, protegidos ou privados, aumentando ainda mais o nível de segurança.

- Herança de código: digamos que você tenha uma classe pessoas com as propriedades nome, email e telefone e precise gerenciar pessoas físicas (com CPF) e jurídicas (com CNPJ). Basta criar uma classe física herdando tudo da classe pessoas e adicionando a propriedade CPF, e fazer a mesma coisa para jurídica.

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

   Porque os requisitos funcionais abordam o que o sistema deve fazer.

   Exemplos:

   1. O sistema deve permitir que cada professor realize o lançamento de notas das turmas nas quais lecionou.
   2. O sistema deve permitir que o aluno realize a sua matrícula nas disciplinas oferecidas em um semestre.

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(“08475051464”);

   contaBancaria1.setDeposito(100.00);

   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.

   Quando uma exceção é **lançada** (*throw*), a JVM entra em estado de alerta e vai ver se o método atual toma alguma precaução ao **tentar** executar esse trecho de código. Já o return serve para encerrar um método, retornando o que a assinatura disse que iria retornar

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

   Public interface ContaBancaria

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

   Com o polimorfismo vamos ter um controle maior sobre as subclasses sem ter que nos preocupar especificamente com cada uma delas, pois cada uma terá autonomia para agir de uma maneira diferente. Essas formas, em nosso contexto de programação, são as subclasses/objetos criados a partir de uma classe maior, mais geral, ou abstrata.
   Polimorfismo é a capacidade que o Java nos dá de controlar todas as formas de uma maneira mais simples e geral, sem ter que se preocupar com cada objeto especificamente.

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

Porque na interface existe apenas assinaturas de métodos, sendo assim eles serão implementados pela classe que compôs da maneira que a classe quiser. E em classes abstratas já podem ter métodos implementados não podendo ser modificados.

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?

O forte acoplamento pode deixar os objetos mais seguros e difíceis de serem modificados, mas ao mesmo tempo perde a coesão, dificultando a manutenção e expansão do software.

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

A decomposição poderia, por exemplo, esconder uma senha de algo, codificando de alguma maneira.

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 que ela tem como dependência, devemos minimizar isso porque devemos ter um abaixo acoplamento em nosso projeto. Quanto mais responsabilidades mais acoplamento terá o projeto.

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.