1-

Analise: modela o problema, investigando quais são. Analise consiste em toda informação feitas com o cliente

Projeto: modela as soluções para os problemas.

2- A vantagem é abstração, fazendo com que o codigo fique menor, mais compreensivel, e de fácil reutilização, e maior facilidade de manutenção.

3- Atender uma exigencia, escopo, problema com clareza,. Identificação de requisito, dependencia de requisito, entrada e saida, o fluxo de interação com o usuario.

4-

public static void main(String args[]){

ContaBancaria conta = new ContaBancaria(11111111111);

conta.depositar(100);

conta.imprimirSaldo();

}

5- É padrão de projeto. A principal vantagem é que você não tem necessidade de entender toda a estrutura, pode colocar qualquer objeto lá dentro e ele vai varrer de forma simples;

6-

Public abstract class ContaBancaria{

private String nome;

private String cpf;

private double saldo;

private int numConta;

public ContaBancaria(String nome, String cpf, double saldo, int numConta){

this.nome = nome;

this.cpf = cpf;

this.saldo = saldo;

this.numConta = numConta;

}

}

7 - o “return” retorna o valor de um atributo, que normalmente está como privado. E o “throw” manda uma exceção ser lançada.

8-

**public** **abstract** **class** ContaBancaria {

**private** **int** numConta;

**private** String nome;

**private** **double** saldo;

**public** **int** getNumConta() {

**return** numConta;

}

**public** **void** setNumConta(**int** numConta) {

**this**.numConta = numConta;

}

**public** String getNome() {

**return** nome;

}

**public** **void** setNome(String nome) {

**this**.nome = nome;

}

**public** **double** getSaldo() {

**return** saldo;

}

**public** **void** setSaldo(**double** saldo) {

**this**.saldo = saldo;

}

}

**public** **class** ContaCorrente **extends** ContaBancaria{

//atributos adicionais/modificados proprios da conta corrente.

}

**public** **class** ContaPoupanca **extends** ContaBancaria{

//atributos adicionais/modificados proprios da conta poupança.

}

9- Vantagem: Permite que um mesmo atributo tenha um valor diferente quando instanciado, assim, fazendo com que ocupe menos memória, deixando o código com maior reaproveitamento e dando uma otimizada no desempenho dele.

Desvantagem: Dependência de herança.

10- Pois se pode usar mais de uma interface para uma classe, além do fato, que todos os métodos de uma interface serão obrigados a serem implementados, não deixando nenhum “sem uso” implicitamente.

11- Sei não..

12- Desvantagens: Difícil manutenção, altas taxas de bugs.

Vantagens: Não sei.

13- métodos com funções complementares, ex: um “calcularImposto()” em uma classe “venda”.

14- É colocar apenas os métodos que são realmente necessários na classe, desacoplando o código.

15 não sei