1)

a. Verdadeiro
b. Verdadeiro
c. Verdadeiro
d. Falso. Os métodos declarados como private só podem ser acessados pela classe em que foram declarados.

2) Artefato, Livro, Capítulo e Página são classes.

A classe Artefato tem como atributo nome (tipo String) e como método imprimeNome (com retorno do tipo void e sem entradas);

A classe Livro tem como atributo autor (do tipo String) e como método imprimeNome (com retorno do tipo void);

A classe Capítulo tem como atributo numero (do tipo int) e numeroDePags (do tipo int);

A classe Página tem como atributo numero (do tipo int);

A classe Livro é uma herança de Artefato;

A classe Capítulo é uma composição de Livro com multiplicidade um ou muitos para um;

A classe página é dependência de Capítulo com multiplicidade um ou muitos para um;

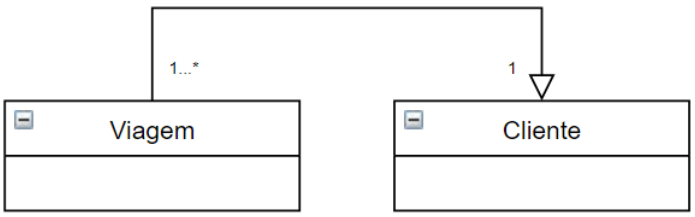
A visibilidade de todos os atributos e métodos é public, pois não estão determinadas;

Código: considerando que o método imprimeNome() é diferente para cada classe e que foi feito um array do tipo Artefato que também tem elementos do tipo Livro, quando o método for chamado ele será executado de acordo com a classe a qual pertence.

```
public class Principal {
    public static void main(String [] args) {
        Artefato a1[] = new Artefato[3];

        a1[0] = new Livro("Orgulho e Preconceito", "Jane Austen");
        a1[1] = new Livro("Cinderela", "Irmãos Grinm");
        a1[2] = new Artefato("Santo Graal");

        for(int i=0; i<a1.length; i++) {
            a1[i].imprimeNome();
        }

    }
}
```

3)

v. Gráfico UML:



vi. Para implementar o relacionamento entre Viagem e Cliente no Java poderíamos utilizar Herança, já que herança é uma relação de um ou muitos para um.

vii. O relacionamento entre Viagem e ViagensAereas é de Herança, sendo Viagem a superclasse e ViagensAereas a subclasse, visto que para se cadastrar uma viagem aérea precisamos de todos os dados de uma viagem qualquer com mais alguns.

viii. A classe Cliente poderia ser abstrata, visto que para se ter um viagem você precisa ter um cliente o que implicaria em implementar todos os métodos nele inseridos.

Mas acredito que seria mais interessante implementar a classe Viagem como abstrata, considerando que pode haver diferentes tipos de viagens: aérea, de trem, de carro, etc. e todas elas implementariam essa classe abstrata.