2 - Não apresentam, pois, todas as subclasses são exclusivas e não tem possibilidade de se transformar em outra classe, sendo assim não há necessidade de criar várias possibilidades de subclasse.

3 - Todas as heranças são completas e disjuntas. Completa, pois não existe outras subclasses. Disjunta, pois são classes exclusivas uma não pode sobrepor a outra.

4-Quando o objeto é mandado por parâmetro e variável local não estruturada, há ganho de memória, pois assim que o método finaliza aquele objeto instanciado é deslocado da memória, nota-se que também que se ganha em encapsulamento, abaixando o acoplamento, porém, o desempenho cai.

6- Set, pois essa estrutura não permite que coloque objetos repetidos.

7 - public class Cliente {

Set<Animal> animal = new HashSet<> ();

}

public class Animal {

Set<Consulta> consulta = new HashSet<> ();

}

public class Conta {

Set<Exame> exame = new HashSet<> ();

}

public class Historico {

Set<Exame> exame = new HashSet<> ();

Set<Consulta> consulta = new HashSet<> ();

}

8- A estrutura TreeSet compara os elementos e os ordena, os elementos são ordenados à medida que são adicionados na lista.

17 - class sub-class : public super-classe1, public super-classe2, public super-classeN {

public:

atributos-subclasse(int foo, int boo, ... , )

}

22 - Faça um quadro comparativo entre reuso por generalização, realização e delegação, apresentando no mínimo duas vantagens e duas desvantagens para cada um desses conceitos.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Generalização | Realização | Delegação |
| Vantagem | Desempenho | Manutenibilidade | Alto encapsulamento |
| Vantagem | Acoplamento Forte | Revelar as operações de um objeto sem revelar a sua classe. | Dinâmica |
| Desvantagem | Diminuição do encapsulamento dependendo de sua profundidade |  | Baixo Acoplamento |
| Desvantagem | Heranças múltiplas causam maior complexidade do sistema. |  | Baixo Desempenho |