1 - Características principais de um programa orientado a objetos (utilize a nomenclatura VISTA EM CLASSE). Definição de objeto e definição de classe.

O principal objetivo de um programa orientado a objetos é esconder a representação de dados e, para isso, permite ao usuário pensar em termos das características abstratas dos tipos que pretende manipular nos programas.
Assim, a mudança da representação interna dos objetos só pode ser realizada através da interface de funções, que devem garantir a produção de novos estados consistentes.
Temos:

- Método: Descrição dos procedimentos que podem ser aplicados aos dados internos de um objeto.

- Objeto: Um conjunto com um estado interno e um conjunto de procedimentos de interface bem definido, sendo que é possível alterar o estado interno e produzir valores.

- Mensagem: Invocação dos métodos de um objeto.

- Variáveis de Instância: Ambiente com valores do estado interno do objeto associado a nomes

- Classe: Uma entidade capaz de criar objetos de estrutura uniforme, isto é, com o mesmo escopo e métodos.

2 - Quando é recomendado o uso de OO e quando é recomendado o uso de Prog. Funcional?

A programação funcional é mais recomendada quando há uma aproximação matemática nos escopos porque é centrada na definição de funções sem efeito colateral com processamento baseado no fluxo de dados através de outras funções matemáticas, enquanto que a OO é centrada na definição de conjuntos de dados com efeitos colaterais e encapsulamento para garantir a abstração e manutenção da consistência.

3 - Encapsulamento: como é obtida em uma OO

O encapsulamento é uma forma de proteger os dados de um objeto fazendo com que estes só possam ser acessados através de mensagens. Assim, controlamos o uso de atributos e métodos da classe através do 'get' e 'set'. O 'get' recupera os valores e o 'set' atribui valores.

4. - Programação baseada em objetos vs orientada a objetos

Em uma programação baseada em objetos não há o conceito de herança, mas existe o conceito de objetos, métodos, mensagens e variáveis de instância. A orientação a objetos ocorre quando a estrutura da linguagem define categorias de objetos, chamadas classes que são dispostas em uma hierarquia: Objetos de uma classe filha têm como base a mesma estrutura da classe mãe.

5. - Herança: como funciona qual seu objetivo em linguagens orientadas a objeto puras

Assim como foi dito acima, a herança permite que novas categorias sejam criadas baseando-se na classe-mãe. Além disso, em sua forma pura, seu objetivo é a reutilização de código porque proporciona o refinamento de comportamento com granularidade fina e a redefinição do comportamento de métodos.

6. - Orientação a objetos pura: como é vista a aritmética? Como são implementados os condicionais?

Dado que em uma OO pura tudo são objetos, a aritmética e o controle de fluxo são vistas como mensagens. Assim, a aritmética é aplicada em classes primitivas como 1+2 ~~ 1.+(2) e o controle de fluxo é feito com a resolução de mensagens e objetos semelhantes à fechamentos da seguinte forma:

```
<Booleano> ifTrue: <fechamento True> IfFalse: <fechamento False>
```

Na classe primitiva True, o método ifTrue:ifFalse manda a mensagem "value" para o primeiro fechamento e na classe primitiva False, o método ifTrue:ifFalse manda a mensagem "value" para o segundo fechamento.

7 - O que são as "classes primitivas" em smalltalk?

As classes primitivas são as menores unidades de informação necessárias para a modelagem de uma estrutura, portanto, elas representam inteiros, booleanos e reais. Para isso, os códigos não estão escritos em Smalltalk, não há um controle sobre os métodos dessas classes.

8 - Java e Smalltalk vs C++: qual a diferença na implementação de resolução de mensagens? Como isso garante que C++ é muito mais rápido? Qual é o overhead de um envio de mensagens em Smalltalk e Java?

Em Java e Smalltalk é feita a busca dinâmica de métodos, enquanto que em C++ é feita a execução direta do código. Para isso, nos dois primeiros a representação do objeto indica a sua classe e cada classe possui uma associação <nome, código do método>, o que possibilita a procura do método na classe e superclasses. O terceiro representa o endereço dos métodos dentro dos objetos de uma classe, fazendo com que o envio de uma mensagem corresponda à chamada indireta em tempo de execução.
Esse comportamento diferente do C++ garante que ele é mais veloz porque, em comparação com os outros dois, não é necessário que seja feito a busca em cada uma das classes da hierarquia até que se encontre o método.

9 - Java vs C++ Vs Smalltalk: como funcionamos tipos nestas três linguagens? Qual o objetivo da tipagem em Java? E em C++?

O tipo é um conceito abstrato que indica operações que podem ser executadas e podem ou não estar associadas a classes. Em C++ as classes são tipos, em Java os tipos podem ser classes ou interfaces, e em Smalltalk o tipo é um conjunto de métodos.
Em C++ é necessário que um método que pode ser redefinido por uma classe-filha seja categorizado como virtual, a fim de que a tipagem ainda preserve a qualidade de um sistema de classificação eficiente. Enquanto que em Java é necessário que as classes implementem as interfaces para que seja garantido que não hajam erros durante o envio de mensagens, dado que é feita a verificação estática.

10 - No ep3 implementamos métodos primitivos, de maneira semelhante ao que acontece em Smalltalk. Como funciona um método primitivo? Qual a diferença entre ele e um método normal?

Os métodos primitivos ,assim como em Smalltalk, são externos ao interpretador e, portanto, estão definidos em um vetor de 2 índices, respectivamente as 2 mensagens de erros possíveis. Assim, a diferença entre um método normal e um primitivo é a característica de estar definido exteriormente ao interpretador.

11 - Como funciona um envio de mensagem no interpretador? Quais os passos para a construção do ambiente para execução de mensagem?

Primeiramente cria-se uma classe com sua variável de instância e seus dois métodos e em seguida faz-se a instanciação de um objeto dessa classe através do new. Agora, na execução de uma mensagem é feita a chamada do send com um objeto, um método e um parâmetro em seus argumentos. A partir desse momento é feita a busca pela classe do objeto usando o ponteiro contido no objeto no ambiente e verifica se o método usado corresponde aos métodos da classe. A partir disso, executa a expressão com o argumento fornecido se achado e, caso contrário, procura nas classes acima na hierarquia, retornando a mensagem de erro primitiva caso não encontre.

12 - Em Smalltalk em geral erros são apenas o envio de mensagens não implementadas. No nosso interpretador funciona de maneira parecida com smalltalk? O que acontece quando mandamos uma mensagem não implementada? Como podemos, em nosso programa OO, colocar um tratamento alternativo para a mensagem desconhecida?

No interpretador os erros também são envios de mensagens que estão dispostas no vetor de métodos primitivos da classe 'Object', então ocorre de forma semelhante ao smalltalk. Assim, quando procuramos por uma mensagem não implementada, o erro de "MensagemDesconhecida: " é mostrado.
Para criarmos um tratamento alternativo, basta que usemos a propriedade de herança entre classes, pois os métodos primitivos são os métodos da classe Object.

13 - Nosso interpretador contém a classe Object. Como em smalltalk, ela tem métodos. Como você fez para a classe object estar sempre presente?

Todas as classes são filhas da classe Object, que é colocada no ambiente global imediatamente após a inicialização do interpretador. Assim, sempre que um método não é encontrado, os métodos primitivos de erro da classe Object são acionados através de um send.