

Ответьте на вопросы, приведенные ниже. Нарисуйте рисунки, позволяющие вам объяснить ответы.

<p>Перечислите известные вам принципы ООП. Как вы думаете, для чего они необходимы.</p>	<p>Эти принципы - инкапсуляция, наследование и полиморфизм. Все вместе они дают такие преимущества, как:</p> <ul style="list-style-type: none"> - возможность легкого расширения - абстракция от деталей реализации - более удобная декомпозиция (данные и операции отсекаются вместе)
<p>Как вы понимаете, что такое инкапсуляция. Приведите примеры ее использования.</p>	<p>Под инкапсуляцией я понимаю такой способ описания кода, когда в одном месте расположены данные и методы их обработки, но часть из них скрыта для внешнего доступа. Например в последнем также я сделал открытый только метод calculate, а другие вспомогательные методы сделал private, так как это минимизирует для подзатаски.</p>
<p>Что такое наследование? Какие различия вы видите, если сравнивать наследование и реализацию интерфейсов.</p>	<p>Наследование - это механизм, который позволяет обобщить код. При наследовании одного класса другим классом дочерний получает и потом может расширять поля и методы базового класса. А при реализации интерфейса мы только задаем модели извлечения (говорим, что этот класс должен выполнять такие-то методы)</p>
<p>Как вы понимаете, что такое полиморфизм? Какие преимущества дает использование полиморфизма?</p> <p>Попробуйте вообразить, как писался бы код, если бы полиморфизма не было.</p>	<p>Это способность ссылок базового типа ссылаться на объекты производных типов. В паре с наследованием он дает такие преимущества в виде гибкости и расширяемости (привязка к интерфейсам). Если бы не было полиморфизма, то мог бы быть местко привязано к конкретным типам. А возможность расширения без изменения и предоставить нечто:</p>

А если еще и перегрузку считать частным случаем полиморфизма, то нам бы не хватило французской машины

<p>Объясните, как вы понимаете, что такое статическое(раннее) и динамическое(позднее) связывание.</p>	<p>Раннее связывание происходит рано, во время компиляции на основе типа ссылочной переменной.</p> <p>Позднее происходит в runtime с использованием конкретных объектов</p>
<p>Напишите (нарисуйте) алгоритм, описывающий порядок вызова конструкторов при создании объектов производного класса. (Постарайтесь предусмотреть все случаи)</p>	<p>B extends A</p> <ol style="list-style-type: none"> 1) статические поля A 2) стат. блок A 3) стат. поля B 4) стат. блок B 5) нестат. поля A 6) нестат. блок A 7) конструктор A 8) нестат. поля B 9) нестат. блок B 10) конструктор B
<p>Как вы думаете, зачем необходимо переопределение методов? Приведите примеры переопределенных методов. Может ли при переопределении метода меняться тип возвращаемого значения?</p>	<p>Это необходимо для эффективной реализации полиморфизма. Класс потомка может хотеть по-своему реализовать метод базового класса, при этом вызвать переопределенный метод можно, используя ссылку базового типа. Примеры: <code>toString()</code>, <code>equals()</code>, <code>hashCode()</code>.</p> <p>Тип возвращаемого значения может изменяться на портпл</p>
<p>Объясните, как вы понимаете смысл фразы "ссылка базового типа может ссылаться на объекты производных типов" и "объект подкласса может быть использован везде, где используется объект его суперкласса."</p>	<p>Если класс <code>Sword</code> extends <code>Weapon</code>, то мы можем написать:</p> <p><code>Weapon w = new Sword();</code></p> <p>И если, например, в методе в качестве параметра нужен <code>Weapon</code>, то мы можем туда передать и <code>Weapon</code>, и <code>Sword</code> и любой другой подкласс</p>

подкласс

Попробуйте порассуждать на тему, наследуются ли статические методы.

Если `class B extends A`, то по ссылке `B` мы получим доступ к статическим методам класса `A`. Не зря же, можно же тут говорить о наследовании, ведь если мы в `B` опишем метод с такой же сигнатурой, то произойдет скретчинг, а не переопределение.

Как вы понимаете, то такое перегруженные методы?
Приведите примеры перегруженных методов.

Это методы с одинаковым именем, но в разном типом и/или количеством параметров. Например, метод `printf` из `System.out`

Как вы думаете, зачем в языке Java надо применять ключевое слово `final`. Что может быть `final` в Java.

`final` метод не может быть перегружаться
`final` класс не может наследоваться
`final` переменная не может быть изменена

Перечислите методы класса `Object`, которые вы знаете. Расскажите для чего они предназначены и как работают.

`clone` — создает копию объекта
`equals` — определяет равенство объектов (изначально по ссылкам, надо переопределять)
`getClass` — возвр. объект `Class`
`hashCode` — возвр. хэшкод (надо переопределять)
`toString` — возвращает строковое представление объекта (многие переопределены)

Также есть метод `finalize` и методы для работы с потоками.



Для чего следует применять метод equals(). Знаете ли вы "где" в Java метод equals() применяется неявно?
Перечислите правила переопределения метода equals()

Призначається для супернення двох об'єктів

Правил:

- реордексивність (рівні самому собі)
- симетричність (якщо $a=b$, тоді $b=a$)
- транзитивність (якщо $a=b$, $b=c$, тоді $a=c$)
- непротиворечливість (одинаково рахується при многократному використанні)
- не може бути ссылка (чищо не рівно нічого)

Метод equals() неявно може використовуватися в, например, Hash Map для про-

Что такое хэш-код? Что такое хэш-код объекта? Объясните, почему хэш-коды двух различных объектов могут совпасть. Перечислите известные вам правила переопределения метода hashCode()

Хэш-код — это число, верное для равенства

Хэш-код объекта — это число, восточно-такое на основании состояния объекта (его полей). Хэш-функция гарантирует одинаковое хэш-коды одинаковых объектов, но не дает гарантии обратной (мог ограничено int диапазоном). Следует переопределять всегда, когда переопределены equals(). Значение хэш-кода не должно меняться, если объект не был изменен.

Расскажите, когда применяется метод toString() и как его необходимо переопределять

Метод `toString` применяется, когда мы хотим получить текстовое представление объекта, например:
`System.out.print("my Obj");`

При переопределении нужно укачивать значение для всех полей объекта

Что представляет собой перечисления в Java и чем по сути является каждый элемент перечисления.

Это класс спечаленного вида, наследник от класса Enum. Элементы перечисления — это по сути набор логически связанных констант

Что, кроме элементов перечисления, можно определить в перечислении?	Как и обычные массивы, перечисления могут определять конструкторы, поля и методы. Могут реализовываться интерфейсами.
Как вы думаете, можно ли (а если можно, то нужно ли) в перечислении переопределять методы equals(), hashCode() и toString()	equals и hashCode нельзя, они final в классе Enum. toString можно, но не нужно, так как в классе Enum они переопределены для возврата одинаковых значений