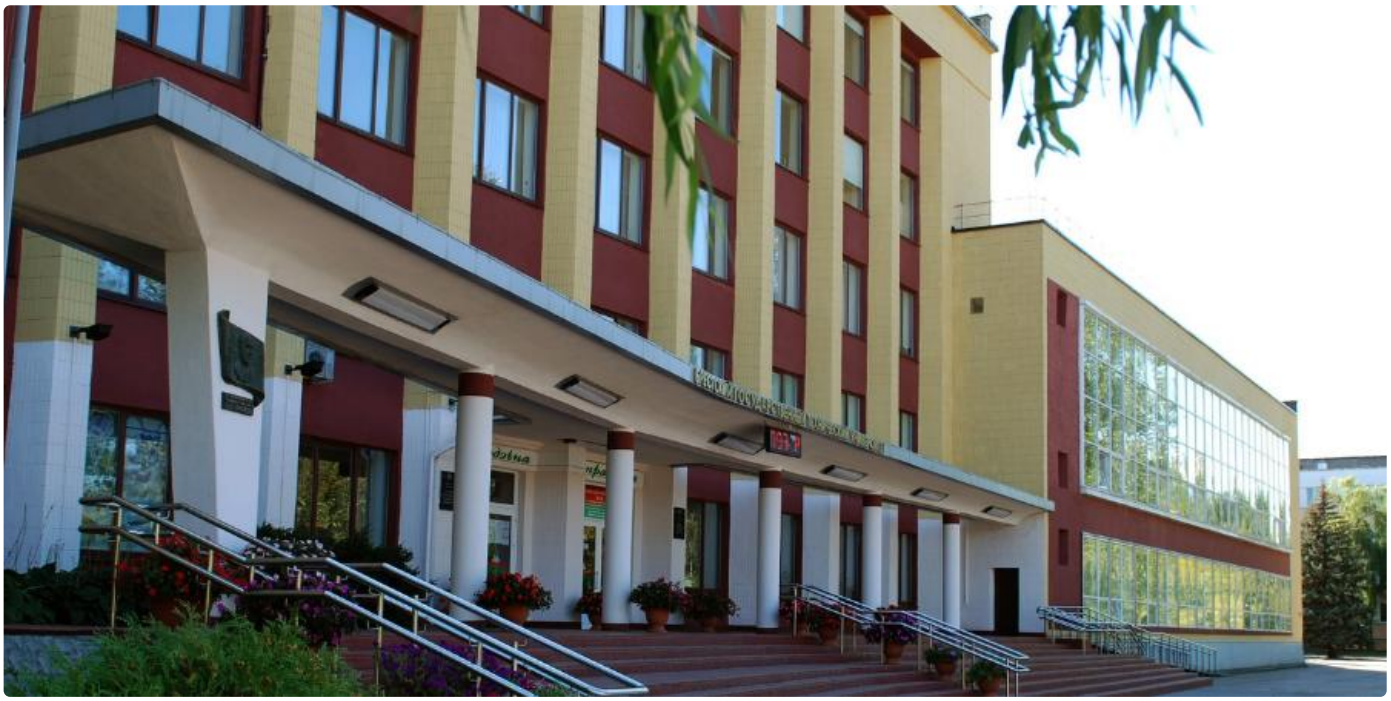
СПП



Мой вариант теста на СПП. Чтобы сдать, необходимо ответить на 12 вопросов (это 4). Может хватит и 11 (там округляется до 4, но это уже по настроению преподавателя). Тем, кто получит 10, будет готов сертификат

Особая благодарность тем, кто помогал мне:
ПО-4
1. Тупик Денис (я)
2. Луд Алексей
3. Воробей Анастасия
4. Галанин Павел
5. Коташевич Станислав
ПО-5
1. Нерода Александра

Пострадал больше всего - Иваненко Иван (ПО-4)

Инструкция к тесту

Количество вопросов: 30
Количество минут на прохождение: 30
Количество правильных ответов в вопросе: 1

Заполните форму

Фамилия

Имя

Группа

Автор: Тупик Денис Леонтьевич

Источник: Ссылка на вопросы

СПП

1 1 из 288

Java. Общие сведения

Just-In-Time-компилятор позволяет

- [x] технология увеличения производительности программных систем, использующих байт-код, путём компиляции байт-кода в машинный код или в другой формат непосредственно во время работы программы.
- [ ] программа, переводящая написанный на языке программирования текст в набор машинных кодов.
- [ ] построчный анализ, обработка и выполнение исходного кода программы или запроса,
- [ ] технология уменьшения производительности программных систем, использующих байт-код, путём компиляции байт-кода в машинный код или в другой формат непосредственно во время работы программы.

2 2 из 288

Java. Общие сведения

Для разработки программ на языке Java необходима установка

- [x] JDK
- [ ] SDK
- [ ] .NET
- [ ] JRE

3 3 из 288

Java. Общие сведения

Запуск Java-программы производится

- [x] java Main arg0 arg1 arg2
- [ ] java arg0 arg1 arg2
- [ ] javac Main arg0 arg1 arg2
- [ ] javac arg0 arg1 arg2

4 4 из 288

Java. Общие сведения

Использование языковой виртуальной машины позволяет добиться

- [x] Кроссплатформенности
- [ ] Монолитности
- [ ] Масштабируемости
- [ ] Совместимоти

5 5 из 288

Java. Общие сведения

Компиляция исходного кода производится

- [x] javac
- [ ] java
- [ ] javab
- [ ] jar

6 6 из 288

Java. Общие сведения

Особенностью языка программирования Java является

- [ ] Отсутствие множественного наследования
- [ ] Интерфейсы
- [ ] Перечисления
- [ ] Поддержка механизма обработки исключений
- [ ] Обобщенные (generics) классы
- [ ] Аннотации
- [ ] Средства параллельного программирования
- [ ] Поддержка лямбда-выражений (начиная с 8 в.)
- [x] Всё вышеперечисленное

7 7 из 288

Java. Общие сведения

Пакеты служат для

- [x] для работы с пространством имен, так и для ограничения видимости.
- [ ] для работы с функциями и классами.
- [ ] для работы с методами и полями.
- [ ] для работы с интерфейсами, перечислениями, коллекциями, объединениями и потоками.

8 8 из 288

Java. Общие сведения

Сигнатура функции main имеет вид

- [x] main(String[] args)
- [ ] public static void main(String[] args)
- [ ] main(String arg)
- [ ] public static void main(String arg)

9 9 из 288

Java. Общие сведения

Синтаксис языка программирования Java схож с синтаксисом языка

- [x] C#
- [ ] Assebbler
- [ ] C++
- [ ] Go

10 10 из 288

Java. Общие сведения

Структурными частями простейшего Java-проекта являются

- [ ] src - исходный код
- [ ] lib - библиотеки
- [ ] res - прочие ресурсы
- [ ] test - исходный код тестов
- [x] всё перечисленное

11 11 из 288

Java. Общие сведения

Точкой входа в любую Java-программу является

- [x] main(String[] args)
- [ ] main(String arg)
- [ ] int main(String arg)
- [ ] int main(String[] args)

12 12 из 288

Java. Общие сведения

Язык программирования Java является

- [x] • Языком высокого уровня
  • Императивным
  • Со строгой статической типизацией
  • Объектно-ориентированный
- [ ] • Языком высокого уровня
  • Императивным
  • Без строгой статической типизации
  • Функциональным
- [ ] • Языком низкого уровня
  • Императивным
  • Со строгой статической типизацией
  • Объектно-ориентированный
- [ ] • Языком низкого уровня
  • Императивным
  • Без строгой статической типизации
  • Функциональным

13 13 из 288

Преобразование целочисленных чисел и чисел с плавающей точкой

В выражении, содержащем переменные типа byte, short, char происходит автоматическое повышение типа до

- [x] int
- [ ] byte
- [ ] short
- [ ] char

14 14 из 288

Преобразование целочисленных чисел и чисел с плавающей точкой

В выражении, содержащем переменные типа double и long, происходит автоматическое повышение типа до

- [x] double
- [ ] long
- [ ] float
- [ ] short

15 15 из 288

Преобразование целочисленных чисел и чисел с плавающей точкой

В выражении, содержащем переменные типа float и double, происходит автоматическое повышение типа до

- [x] double
- [ ] float
- [ ] long
- [ ] short

16 16 из 288

Преобразование целочисленных чисел и чисел с плавающей точкой

В выражении, содержащем переменные типа float, происходит автоматическое повышение типа до

- [x] float
- [ ] int
- [ ] double
- [ ] short

17 17 из 288

Преобразование целочисленных чисел и чисел с плавающей точкой

В выражении, содержащем переменные типа int и long, происходит автоматическое повышение типа до

- [x] long
- [ ] int
- [ ] short
- [ ] unsigned int

18 18 из 288

Преобразование целочисленных чисел и чисел с плавающей точкой

К классам-оберткам над элементарными типами данных не относится

- [x] int
- [ ] Integer
- [ ] Float
- [ ] Double

19 19 из 288

Преобразование целочисленных чисел и чисел с плавающей точкой

К типам с плавающей запятой относится тип

- [x] float , double
- [ ] float , int
- [ ] long , double
- [ ] long , int

20 20 из 288

Преобразование целочисленных чисел и чисел с плавающей точкой

К целочисленным типам данных Java не относится тип

- [x] double
- [ ] int
- [ ] long
- [ ] short

21 21 из 288

Ссылки

В Java ссылки всегда передаются

- [x] объекты
- [ ] структуры
- [ ] аргументы
- [ ] переменные

22 22 из 288

Ссылки

Значение null используется в Java

- [x] значение по умолчанию для ссылочных типов
- [ ] значение по умолчанию для нессылочных типов
- [ ] значение по умолчанию для примитивных типов
- [ ] значение по умолчанию для Ивана Леонидовича

23 23 из 288

Ссылки

Объект может быть передан по ссылке

- [x] но не влияет на сам объект
- [ ] но влияет на сам объект
- [ ] верны два варианта
- [ ] ошибочны два варианта

24 24 из 288

Ссылки

Объект называется достижимым, если

- [x] если на него ссылается другой достижимый объект.
- [ ] если на него не ссылается другой достижимый объект.
- [ ] если на него не ссылается другой недостижимый объект.
- [ ] если на него ссылается другой недостижимый объект.

25 25 из 288

Ссылки

Преобразование типа переменной подкласса к типу суперкласса

- [x] происходит автоматически
- [ ] происходит вручную
- [ ] не происходит
- [ ] происходит по воле Иваненко

26 26 из 288

Ссылки

Преобразование типа переменной суперкласса к типу подкласса

- [x] требует явного преобразования
- [ ] требует неявного преобразования
- [ ] не требует преобразования
- [ ] данное преобразование запрещено в Java

27 27 из 288

Ссылки

Приведение ссылочных типов возможно

- [x] если классы в одной иерархии
- [ ] если классы в разных иерархиях
- [ ] если классы не находятся в иерархиях
- [ ] привидение невозможно в Java

28 28 из 288

Ссылки

Прямое присваивание одной ссылке значения другой ссылки приведет к тому

- [x] что происходит переприсваивание ссылки
- [ ] что не происходит переприсваивание ссылки
- [ ] ничего не происходит
- [ ] Я Ваня Иваненко! Чего ты пристал?

29 29 из 288

Символы и строки в Java

Для представления символов в Java используется кодировка

- [x] UTF-16
- [ ] UTF-8
- [ ] UTF-4
- [ ] UTF-2

30 30 из 288

Символы и строки в Java

Изменение отдельного символа строки

- [x] невозможно, если не использовать StringBuilder
- [ ] невозможно, если не использовать StringBuffer
- [ ] невозможно, если не использовать String
- [ ] невозможно в любом случае

31 31 из 288

Символы и строки в Java

Получение отдельного символа строки

- [x] String.charAt()
- [ ] String.getChar()
- [ ] string[i]
- [ ] String.get()

32 32 из 288

Символы и строки в Java

При именовании переменных в Java не допускаются

- [ ] ключевые или зарезервированные слова, пробелы
- [ ] не начинать с цифр, может состоять из букв (Unicode)
- [ ] цифр и символа подчеркивания «_»
- [x] всё выше перечисленное не допускается

33 33 из 288

Символы и строки в Java

Строковые литералы в Java являются

- [x] набор символов в двойных кавычках
- [ ] набор символов в одинарных кавычках
- [ ] набор символов в квадратных кавычках
- [ ] набор символов в фигурных кавычках

34 34 из 288

Символы и строки в Java

Тип char относится к

- [x] целочисленным
- [ ] строковым
- [ ] вещественным
- [ ] объектным

35 35 из 288

Символы и строки в Java

У переменных типа String нет встроенных функций, поддерживающих следующую операцию над строками

- [x] parseInt() : перевод строки в число
- [ ] concat() : объединяет строки
- [ ] equals() : сравнивает строки с учетом регистра
- [ ] trim() : удаляет начальные и конечные пробелы

36 36 из 288

Символы и строки в Java

Пул Java String это

- [x] это множество строк в кучи
- [ ] это множество строк в стеке
- [ ] это множество строк в очереди
- [ ] это множество строк в массиве

37 37 из 288

Символы и строки в Java

Основное различие между StringBuffer и StringBuilder

- [x] StringBuffer потокобезопасный и синхронизированный
- [ ] StringBuffer не потокобезопасный и не синхронизированный
- [ ] StringBuffer потокобезопасный и не синхронизированный
- [ ] StringBuffer не потокобезопасный и синхронизированный

38 38 из 288

Символы и строки в Java

StringBuffer является

- [x] потокобезопасный и синхронизированный
- [ ] не потокобезопасный и не синхронизированный
- [ ] потокобезопасный и не синхронизированный
- [ ] не потокобезопасный и синхронизированный

39 39 из 288

Преобразование строчных типов в число

Выберите метод, не преобразующий строку

- concat()
- replace()
- split()
- ✓ всё преобразует в строку

40 40 из 288

Преобразование строчных типов в число

Для какого потока не является необходимым вызов метода close

- ✓ Если поток открыт в конструкции try-with-resources.
- Если поток закрыт в конструкции try-with-resources.
- Если поток открыт в конструкции try-catch.
- Если поток закрыт в конструкции try-catch.

41 41 из 288

Преобразование строчных типов в число

К стандартным потокам ввода-вывода Java не относится

- FileInputStream
- FileOutputStream
- ObjectOutputStream
- ✓ ClassInputStream

42 42 из 288

Преобразование строчных типов в число

Пакет java.lang является

- ✓ наиболее важным из всех пакетов, входящих в Java API, поскольку включает классы, составляющие основу для всех других классов
- наиболее бесполезным из всех пакетов, входящих в Java API, поскольку включает классы, составляющие основу для всех других классов
- наиболее важным из всех пакетов, входящих в Java REST API, поскольку включает классы, составляющие основу для всех других классов
- наиболее важным из всех пакетов, входящих в Java GRASP, поскольку включает классы, составляющие основу для всех других классов

43 43 из 288

Преобразование строчных типов в число

Пакет java.util является

- пакетом, который используются для работы с набором объектов, взаимодействия с системными функциями низкого уровня, для работы с математическими функциями
- пакетом, который используются для работы с набором объектов, взаимодействия для генерации случайных чисел и манипуляций с датой и временем.
- оба варината верны
- оба варианта - ложь

44 44 из 288

Преобразование строчных типов в число

Строка не может быть преобразована в число типа double применением

- С использованием конструктора
- С использованием метода valueOf класса Double
- С использованием метода parseDouble класса Double
- С использованием метода tryParse класса Double

45 45 из 288

Преобразование строчных типов в число

Строка не может быть преобразована в число типа int применением

- С использованием конструктора
- С использованием метода valueOf класса Integer
- С использованием метода parseInt класса Integer
- С использованием метода tryParse класса Integer

46 46 из 288

Массивы и коллекции

Глубокое копирование массива объектов может быть достигнуто

- Java Object Serialization
- Java Class Serialization
- Java Method Serialization
- Java Serialization

47 47 из 288

Массивы и коллекции

Глубокое копирование массива объектов — это…

- [x] когда объект копируется вместе с объектами, к которым он относится.
- [ ] когда объект копируется вместе с объектами, к которым он не относится.
- [ ] когда объект копируется частично
- [ ] когда объект не копируется

48 48 из 288

Массивы и коллекции

К основному недостатку массива по сравнению с коллекцией относят

- [ ] фиксированную длину
- [ ] индексную адресацию
- [x] оба варианта подходят
- [ ] оба варианта неподходят

49 49 из 288

Массивы и коллекции

Коллекция — это…

- [x] классы, основная цель которых – хранить набор других элементов.
- [ ] объекты, основная цель которых – хранить набор других элементов.
- [ ] классы, основная цель которых – хранить набор своих элементов.
- [ ] объекты, основная цель которых – хранить набор своих элементов.

50 50 из 288

Массивы и коллекции

Основные типы коллекций в Java не включают

- [x] Array
- [ ] Deque
- [ ] List
- [ ] Set

51 51 из 288

Циклы

В языке Java

- [x] есть циклы
- [ ] отсутсвуют циклы
- [ ] Иваненко молодец
- [ ] Тупик круче Иваненко! Ха!

52 52 из 288

Циклы

Выберите оператор, не являющийся оператором цикла

- [x] foreach
- [ ] for
- [ ] while
- [ ] do...while

53 53 из 288

Циклы

Выберите оператор, являющийся оператором ветвления

- [ ] if
- [ ] if...else
- [ ] switch...case
- [x] всё перечисленное

54 54 из 288

Циклы

Выберите оператор, являющийся оператором перехода

- [ ] break
- [ ] continue
- [ ] return
- [x] всё перечисленное

55 55 из 288

Циклы

Оператор switch чаще всего используется

- [x] для организации выбора из множества различных вариантов
- [ ] для организации выбора из 2 вариантов
- [ ] для организации цикла
- [ ] для определённого выхода цикла

56 56 из 288

Циклы

Скомпилируется ли следующий код: int i; int j; (false ? i: j) = 55;

- [ ] да
- [x] нет
- [ ] не знаю
- [ ] Главное, что Иванено ПЛАХОЙ!

57 57 из 288

Циклы

Цикл for в форме for each используется

- [ ] используется для перебора элементов массива
- [ ] используется для перебора элементов коллекции
- [x] оба варианта правильные
- [ ] оба варианта неправильные

58 58 из 288

Классы. Поля, методы и конструкторы

Выберите ложное высказывание относительно конструкторов в классах Java

- [ ] Конструктор — это метод класса, который инициализирует новый объект после его создания.
- [ ] Имя конструктора совпадает с именем класса.
- [ ] У конструкторов нет типа возвращаемого результата.
- [x] Конструкторы нельзя перегружать.

59 59 из 288

Классы. Поля, методы и конструкторы

Выберите ложное высказывание относительно конструкторов в классах Java

- Конструктор — это метод класса, который инициализирует новый объект после его создания.
- Имя конструктора совпадает с именем класса.
- ✓ У конструкторов есть тип возвращаемого результата.
- Конструкторы можно перегружать.

60 60 из 288

Классы. Поля, методы и конструкторы

Для метода с одним параметром, которым является Java-объект, его поля будут передаваться

- ✓ копия ссылки на область памяти, в которой находится объект
- ссылка на область памяти, в которой находится объект
- копия ссылки на область памяти, в которой находится параметр
- ссылка на область памяти, в которой находится параметр

61 61 из 288

Классы. Поля, методы и конструкторы

Методы с модификатором final

- ✓ невозможно изменить
- невозможно наследовать
- верны оба варианта
- оба варианта неверны

62 62 из 288

Классы. Поля, методы и конструкторы

Перегрузка методов — это

- ✓ один из способов поддержки полиморфизма в Java
- один из способов поддержки инкапсуляции в Java
- один из способов поддержки наследования в Java
- всё вышеперечисленное

63 63 из 288

Классы. Поля, методы и конструкторы

Поле с модификатором final не может быть проинициализирована

- [x] дважды
- [ ] однажды
- [ ] никогда
- [ ] нет правильного ответа

64 64 из 288

Классы. Поля, методы и конструкторы

При перегрузке методов

- [x] вы создаете метод с таким же именем, но с другим набором параметров
- [ ] вы создаете метод с таким же именем, и с тем же набором параметров
- [ ] вы создаете метод с другим именем, но с другим набором параметров
- [ ] вы создаете метод с другим именем, но с тем же набором параметров

65 65 из 288

Классы. Поля, методы и конструкторы

Статические методы и поля класса — это элементы…

- [x] к которым можно обращаться используя имя класса
- [ ] к которым нельзя обращаться
- [ ] к которым можно обращаться как обычно
- [ ] к которым нельзя обращаться используя имя класса

66 66 из 288

Классы. Их разновидности

Анонимный класс — это

- [x] локальный класс без имени
- [ ] глобальный класс без имени
- [ ] локальный класс с именем
- [ ] глобальный класс с именем

67 67 из 288

Классы. Их разновидности

Выберите ложное высказывание

- [x] классы имеют множественное наследование
- [ ] классы наследуют много интерфейсов
- [ ] классы могут иметь много конструкторов
- [ ] все высказвания верны

68 68 из 288

Классы. Их разновидности

Выберите ложное высказывание о вложенных классах

- [x] частный случай вложенного класса - внутренний
- [ ] частный случай внутреннего класса - вложенный
- [ ] оба варианта - правда
- [ ] оба варианта - ложь

69 69 из 288

Классы. Их разновидности

Для вызова метода суперкласса в классе-потомке (при наличии в классе-потомке одноименного метода) нужно

- [x] Ключевое слово super
- [ ] Ключевое слово global
- [ ] Ключевое слово base
- [ ] Ключевое слово parent

70 70 из 288

Классы. Их разновидности

Для переопределения метода в классе потомке достаточно

- [x] @Override
- [ ] @Overrage
- [ ] @Moxy
- [ ] @Modifier

71 71 из 288

Классы. Их разновидности

Для указания того, что класс является подклассом используется

- [x] extends
- [ ] implements
- [ ] parents
- [ ] base

72 72 из 288

Классы. Их разновидности

Если в суперклассе есть один конструктор с параметрами, а в классе-потомке определен новый конструктор, не содержащий вызов конструктора суперкласса, то

- [ ] конструкторы вызываются по порядку выведения классов: от суперкласса к подклассу.
- [ ] конструкторы вызываются по порядку выведения классов: от подкласса к сеперклассу.
- [ ] конструкторы вызываются хаотично.
- [x] будет ошибка

73 73 из 288

Классы. Их разновидности

Ключевое слово super нельзя использовать

- [ ] для вызова конструктора суперкласса
- [ ] для обращения к члену суперкласса, скрытому членом подкласса
- [ ] когда перегруженны конструкторы
- [x] можно использовать во всех перечисленных вариантах

74 74 из 288

Классы. Их разновидности

Переопределение методов при наследовании позволяет

- [x] изменять функциональность класса, его поведение
- [ ] изменять имена методов
- [ ] изменять константных значений
- [ ] изменять родительский конструктор

75 75 из 288

Интерфейсы

В интерфейсе все поля являются по умолчанию

- ○ final
- ○ ни final, ни static
- ✓ final и static
- ○ static

76 76 из 288

Интерфейсы

В интерфейсе все функции являются по умолчанию

- ○ private
- ✓ public
- ○ protected
- ○ default

77 77 из 288

Интерфейсы

Выберите ложное высказывание об абстрактных классах

- ○ Нельзя инстанцировать объект такого класса
- ✓ Реализация не предоставляется в классах-потомках
- ○ Если классы-потомки не содержат реализации, то они тоже должны быть абстрактными
- ○ Абстрактный класс может содержать полностью реализованные методы

78 78 из 288

Интерфейсы

Выберите ложное высказывание об интерфейсах

- ○ Назначение интерфейсов – предоставлять альтернативу множественному наследованию в том, что касается наследования поведения (т.е. определенных методов)
- ✓ Интерфейс должен содержать описание методов с их реализацией
- ○ Реализация интерфейса производится в классе, в заголовке которого явно указывается необходимость реализации данного интерфейса (ключевое слово implements). Если реализация частичная, то такой класс должен быть абстрактным (abstract)
- ○ Интерфейсы могут формировать иерархии (расширение интерфейсов)

79 79 из 288

Интерфейсы

Динамическая диспетчеризация методов позволяет

- [x] ссылочной переменной суперкласса указываем ссылку на подкласс и вызываем переопределенный метод.
- [ ] нессылочной переменной суперкласса указываем ссылку на подкласс и вызываем переопределенный метод.
- [ ] ссылочной переменной суперкласса указываем ссылку на родительский класс и вызываем переопределенный метод.
- [ ] нессылочной переменной суперкласса указываем ссылку на родительский класс и вызываем переопределенный метод.

80 80 из 288

Интерфейсы

Для расширения интерфейса используется ключевое слово

- [x] extends
- [ ] implements
- [ ] base
- [ ] parents

81 81 из 288

Интерфейсы

Для создания классов, которые не могут иметь потомков, применяется

- [x] final
- [ ] static
- [ ] sealed
- [ ] const

82 82 из 288

Интерфейсы

Для указания того, что класс реализует интерфейс используется

- [ ] extends
- [x] implements
- [ ] base
- [ ] parents

83 83 из 288

Интерфейсы

Интерфейс — это

- [x] это абстрактный тип, используемый для описания поведения, которое должны реализовать классы.
- [ ] это конкретный тип, используемый для описания поведения, которое должны реализовать классы.
- [ ] это абстрактный тип, используемый для описания поведения, которое должны реализовать структуры.
- [ ] это конкретный тип, используемый для описания поведения, которое должны реализовать структуры.

84 84 из 288

Интерфейсы

Класс с модификатором final

- [ ] запрещает наследование
- [ ] все методы неявно с модификтором final
- [x] оба варианта верны
- [ ] оба варианта неверны

85 85 из 288

Интерфейсы

default-методы в интерфейсах это

- [x] методы, реализованные в интерфейсе
- [ ] методы, реализованные в классе
- [ ] методы, реализованные в структуре
- [ ] методы, реализованные в перечислении

86 86 из 288

Интерфейсы

Статические методы в интерфейсах это

- [ ] методы, которые нельзя переопределить в объекте
- [x] методы, которые нельзя переопределить в классе
- [ ] методы, которые нельзя переопределить в структуре
- [ ] методы, которые нельзя переопределить в перечислении

87 87 из 288

Перечисления

Выберите ложное высказывание о перечислениях

- Перечисления – списки именованных констант
- Перечисления в Java представляют собой классы (могут содержать в себе конструкторы, методы, поля данных)
- Перечисления не могут быть суперклассами и не могут наследоваться от других классов
- ✓ Перечисления могут быть анонимными

88 88 из 288

Перечисления

Выберите ложное высказывание о перечислениях

- Перечисления – списки именованных констант
- Перечисления в Java представляют собой классы (могут содержать в себе конструкторы, методы, поля данных)
- ✓ Перечисления могут быть суперклассами и могут наследоваться от других классов
- Все высказывания верны

89 89 из 288

Перечисления

Выберите неверное утверждение о пакетах

- пакеты - это некий контейнер, который используется для того, чтобы изолировать имена классов
- пакеты - это механизм, который служит как для работы с пространством имен, так и для ограничения видимости
- пакеты подключаются директивой import
- ✓ пакеты - это механизм подключения файлов и БД

90 90 из 288

Перечисления

Ключ -classpath при запуске утилиты java используется для

- ✓ определения списка каталогов, JAR-файлов и ZIP-архивов для поиска файлов классов
- нахождения заголовочного файла
- нахождения пакета с конфигурацией
- определения проекта, в котором хранятся всё необходимое для запуска проекта

91 91 из 288

Перечисления

Ключ -d при запуске утилиты javac используется для

- [x] указания каталога назначения для файлов классов
- [ ] указания каталога назначения для JRE-файлов
- [ ] указания каталога назначения для JAR-файлов
- [ ] указания каталога назначения для файлов сборки проекта

92 92 из 288

Перечисления

Пакеты в Java подключаются с помощью директивы

- [x] import
- [ ] package
- [ ] using
- [ ] include

93 93 из 288

Перечисления

Перечисление имеет предопределенный метод

- [x] public static тип_перечисления[] values()
- [ ] public static тип_перечисления[] valuesOf(String строка)
- [ ] public static тип_перечисления[] indexes()
- [ ] public static тип_перечисления[] indexesOf(String строка)

94 94 из 288

Перечисления

Перечисление имеет предопределенный метод

- [x] public static тип_перечисления valuesOf(String строка)
- [ ] public static тип_перечисления indexOf(String строка)
- [ ] public static тип_перечисления value()
- [ ] public static тип_перечисления index()

95 95 из 288

Исключения

Выберите ложное утверждение об исключениях

- Обработка исключений – механизм языка программирования Java, применяемый для корректной обработки т.н. исключительных ситуаций (деление на ноль, выход за пределы диапазона индексов массива, открытие несуществующего файла и т.д.
- Включает блоки: try, catch, finally
- Существует 2 вида исключений : проверяемые/непроверяемые
- ✓ Сущетсвует 4 класса: Exception, RuntimeException, Error и RuntimeError

96 96 из 288

Исключения

Выберите признак проверяемого исключения

- Блок catch должен присутствовать в той же функции, где генерируется исключение.
- Если catch отсутствует, то заголовок функции должен содержать оператор throws с указанием типа исключения для обработки этого исключения в вызывающей функции.
- ✓ верны оба варианта
- ошибочны оба варинта

97 97 из 288

Исключения

Выход за пределы индексации массива относится к типу исключения

- ✓ IndexOutOfBoundException (RuntimeException)
- IndexOutOfBoundException (RuntimeError)
- OutOfMemoryError (RuntimeException)
- OutOfMemoryError (RuntimeError)

98 98 из 288

Исключения

К классам исключений не относится

- Exception используется для перехвата пользовательских исключений
- RuntimeException используется для стандартных исключений времени выполнения (деление на ноль, выход за диапазон индексации и т.д.)
- Error используется для внутрисистемных исключений
- ✓ RuntimeError используется для внитрисистемных исключений времени выполнения (бесконечный цикл, окончание времени ожидания)

99 99 из 288

Исключения

К основным методам класса Throwable не относится

- void printStackTrace()
- void printStackTrace(PrintStream поток)
- StackTraceElement [] getStackTrace()
- ✓ void setStackTrace(PrintStream поток)

100 100 из 288

Исключения

Ошибка деления на ноль относится к классу исключения

- ✓ ArithmeticException (RuntimeException)
- NullPointerException (RuntimeException)
- NullResultException (RuntimeException)
- DivideNullException (RuntimeException)

101 101 из 288

Исключения

Ошибка доступа к файлу относится к классу исключения

- ✓ FileNotFoundException (RuntimeException)
- NullPointerException (RuntimeException)
- FileNotExistException (RuntimeException)
- FileNotReadException (RuntimeException)

102 102 из 288

Исключения

К проверяемым типам исключений относится

- ✓ IOException
- RuntimeExeption
- OutOfMemoryError
- StackOverFlowError

103 103 из 288

Исключения

К непроверяемым типам исключений относится

- IOException
- StackOverFlowError
- FileNotFoundExeption
- SocketExeption

104 104 из 288

Регулярные выражения

Знак «*» в регулярном выражении означает

- любое количество экземпляров элемента (в том числе и нулевое)
- любое количество экземпляров элемента (кроме нулевого)
- нулевое количество экземпляров
- нет правильного ответа

105 105 из 288

Регулярные выражения

Знак «.» в регулярном выражении означает

- представляет собой сокращенную форму записи для символьного класса, совпадающего с любым символом
- представляет собой сокращенную форму записи для символьного класса, совпадающего с любым символом, кроме нулевого
- представляет собой сокращенную форму записи для нулевого класса
- представляет собой полную форму записи для символьного класса, совпадающего с любым символом

106 106 из 288

Регулярные выражения

Квантификаторы нужны для

- позволяют задавать количество вхождений символа в строку
- для выделения групп регулярных выражений
- перечня символов которые могут быть (или НЕ могут) на месте данного символа
- для выделения нулевых символов

107 107 из 288

Регулярные выражения

Круглые скобки в регулярном выражении служат для

- позволяют задавать количество вхождений символа в строку
- ✓ для выделения групп регулярных выражений
- перечня символов которые могут быть (или НЕ могут) на месте данного символа
- для выделения нулевых символов

108 108 из 288

Регулярные выражения

Символьный класс определяет

- позволяют задавать количество вхождений символа в строку
- для выделения групп регулярных выражений
- ✓ перечня символов которые могут быть (или НЕ могут) на месте данного символа
- для выделения нулевых символов

109 109 из 288

Регулярные выражения

Укажите, какая строки будут соответствовать указанному регулярному выражению [a-zA-Z]{1}[a-zA-Z\\d\\._] +@([a-zA-Z]+\\.){1,2}((net)|(com)|(org))

- ✓ любой email
- определённый email
- ошибка
- нет правильного ответа

110 110 из 288

Регулярные выражения

Символ «^» в регулярном выражении используется для

- отрицание
- любая цифра
- любой символ
- ✓ новая строка

111 111 из 288

Регулярные выражения

\D в регулярном выражении это

- любая цифра
- любой символ, но не цифра
- любой символ
- символ 0

112 112 из 288

Регулярные выражения

!!!«Жадные» операторы это!!!

- это специальные ограничители, с помощью которых определяется частота появления элемента — символа, группы символов
- это специальные ограничители, с помощью которых определяется частота удаления элемента — символа, группы символов
- это специальные ограничители, с помощью которых определяется частота добавления элемента — символа, группы символов
- это специальные ограничители, с помощью которых определяется частота обновления элемента — символа, группы символов

113 113 из 288

Регулярные выражения

Обратные ссылки используются для

- повторения паттернов поиска без их непосредственного копирования
- повторения паттернов поиска с их непосредственным копированием
- копирования паттернов
- всё перечисленное

114 114 из 288

Регулярные выражения

Интернационализация - это

- процесс разработки приложения такой структуры, при которой дополнение нового языка не требует перестройки и перекомпиляции (сборки) всего приложения.
- адаптация приложения к конкретному языку и региону путем перевода выводимых пользователю текстовых элементов и документации, а также определения данных времени, валют и др., согласно специфике данного региона.
- верны оба варианта
- ошибочны оба варианта

115 115 из 288

Регулярные выражения

Локализация - это

- процесс разработки приложения такой структуры, при которой дополнение нового языка не требует перестройки и перекомпиляции (сборки) всего приложения.
- адаптация приложения к конкретному языку и региону путем перевода выводимых пользователю текстовых элементов и документации, а также определения данных времени, валют и др., согласно специфике данного региона.
- верны оба варианта
- ошибочны оба варианта

116 116 из 288

Регулярные выражения

Класс ResourceBundle используется для

- чтения данных из текстовых файлов свойств (расширение - properties)
- учтения особенности региональных представлений алфавита, символов, чисел и дат
- верны оба варианта
- ошибочны оба варианта

117 117 из 288

Регулярные выражения

Класс Locale используется для

- чтения данных из текстовых файлов свойств (расширение - properties)
- учтения особенности региональных представлений алфавита, символов, чисел и дат
- верны оба варианта
- ошибочны оба варианта

118 118 из 288

Регулярные выражения

Для форматирования чисел в Java используется класс

- DecimalFormat
- DecimalString
- DecimalNumber
- всё перечисленное

119 119 из 288

Сериализация

Восстановление объекта при сериализации производится

- нужно упаковать InputStream в ObjectInputStream и вызвать метод readObject()
- необходимо выполнить обработку исключения ClassNotFoundException
- ✓ всё перечисленное
- из системы, в которой происходит восстановления объекта, должен быть доступен файл класса.

120 120 из 288

Сериализация

Выберите истинное утверждение о сериализации

- ✓ Сохранены и восстановлены абсолютно все поля, даже те у которых указан тип доступа private и protected
- Serializable работает только с методами
- Новый объект будет создан только если явно есть конструктор без параметров
- Передача сериализованного объекта по сети, сохранение его на диске и т.д. запрещена

121 121 из 288

Сериализация

Для указания того, что во время сериализации объекта некоторое поле нужно игнорировать, используется модификатор

- ✓ transient
- package
- stream
- serialization

122 122 из 288

Сериализация

К несериализуемому системному типу относится

- ✓ Object
- Task
- InputStream
- WebSocket

123 123 из 288

Сериализация

К сериализуемым системным типам относится

- [x] нет правильного ответа
- [ ] Thread
- [ ] OutputStream
- [ ] Socket

124 124 из 288

Сериализация

Необходимым условием для сериализации объектов класса является

- [x] реализация интерфейса java.io.Serializable
- [ ] реализация интерфейса java.util
- [ ] ключевого слова transient
- [ ] специального пакета Serializable

125 125 из 288

Сериализация

Сериализация — это

- [x] это процесс перевода структуры данных или состояния объекта в формат, который может быть сохранен и восстановлен потом в другой компьютерной среде
- [ ] это особый процесс работы с переменными, который может быть сохранен и восстановлен потом в другой компьютерной среде
- [ ] это процесс работы с облачным хранилищем
- [ ] это процесс перевода структуры данных или состояния объекта в формат, который может быть удалён и стёрт потом в другой компьютерной среде

126 126 из 288

Сериализация

Сохранение объекта при сериализации производится

- [x] с помощью класса java.io.ObjectOutputStream
- [ ] с помощью класса java.io.ObjectInputStream
- [ ] с помощью класса java.io.OutputStream
- [ ] с помощью класса java.io.InputStream

127 127 из 288

Сериализация

Десериализация объекта невозможна если

- [x] у родительского несериализуемого класса не будет конструктора без параметров
- [ ] у родительского сериализуемого класса не будет конструктора без параметров
- [ ] у родительского несериализуемого класса будет конструктора без параметров
- [ ] у родительского сериализуемого класса будет конструктора без параметров

128 128 из 288

Сериализация

Сериализацию классов-наследников сериализуемого типа можно запретить, если

- [x] имеющие модификатор static
- [ ] имеющие модификатор final
- [ ] имеющие модификатор global
- [ ] все варианты верны

129 129 из 288

Общие методы для всех объектов

К методам, общим для всех объектов не относится

- [ ] clone()
- [ ] equals()
- [ ] toString()
- [x] compare()

130 130 из 288

Общие методы для всех объектов

Функция clone()

- [x] создаёт новый объект, не отличающий от клонируемого
- [ ] создаёт новый объект и при этом ссылается на объект-исходник
- [ ] сравнивает вызывающий объект с объектом, переданным в качестве параметра
- [ ] возвращает хеш-код, связанный с вызывающим объектом

131 131 из 288

Общие методы для всех объектов

Функция compareTo()

- [x] сравнивает вызывающий объект с объектом, переданным в качестве параметра
- [ ] создаёт новый объект, не отличающий от клонируемого
- [ ] возвращает хеш-код, связанный с вызывающим объектом
- [ ] возвращает строку, описывающий объект

132 132 из 288

Общие методы для всех объектов

Функция equals()

- [x] сравнивает ссылки
- [ ] сравнивает вызывающий объект с объектом, переданным в качестве параметра
- [ ] возвращает хеш-код, связанный с вызывающим объектом
- [ ] создаёт новый объект и при этом ссылается на объект-исходник

133 133 из 288

Общие методы для всех объектов

Функция hashCode()

- [x] возвращает хеш-код, связанный с вызывающим объектом.
- [ ] сравнивает вызывающий объект с объектом, переданным в качестве параметра
- [ ] создаёт новый объект, не отличающий от клонируемого
- [ ] создаёт новый объект и при этом ссылается на объект-исходник

134 134 из 288

Общие методы для всех объектов

Функция toString()

- [x] возвращает строку, описывающий объект
- [ ] сравнивает вызывающий объект с объектом, переданным в качестве параметра
- [ ] возвращает хеш-код, связанный с вызывающим объектом
- [ ] сравнивает ссылки

135 135 из 288

Общие методы для всех объектов

Функция finalize()

- [x] вызывается Java-машиной у объекта перед тем, как объект будет уничтожен
- [ ] вызывается Java-машиной у объекта перед копированием
- [ ] вызывается Java-машиной у объекта тем, как сравнить строки
- [ ] вызывается Java-машиной у объекта перед, как создать новый объект, не отличающийся от клонируемого

136 136 из 288

UML. Диаграмма классов

Выберите истинное высказывание

- [x] абстрактный классификатор может быть показан после аннотации или под ее именем
- [ ] название абстрактного классификатора показано жирным шрифтом
- [ ] интерфейсы могут соединяться с вариантами использования сплошной линией со стрелкой/сплошной бз стрелки
- [ ] всё ложь

137 137 из 288

UML. Диаграмма классов

Выберите корректный пример отношения композиции

- [x]
```
public class Person {
    //composition has-a relationship
    private Job job;

    //...
}
```
- [ ]
```
public class Person {
    private class Job{

     }
}
```
- [ ]
```
public class Person {
    public class Job{

     }
}
```
- [ ] всё перечисленное

138 138 из 288

UML. Диаграмма классов

!!!Выберите ложное высказывание о спецификации абстрактного класса в UML!!!

- абстрактный классификатор может быть показан с использованием текстовой аннотации {abstract}
- абстрактный классификатор может быть показан после аннотации или под ее именем
- ✓ название абстрактного классификатора показано жирным шрифтом
- всё ложь

139 139 из 288

UML. Диаграмма классов

!!!Выберите ложное высказывание о спецификации интерфейса в UML!!!

- в языке UML интерфейс является классификатором и характеризует только ограниченную часть поведения моделируемой сущности
- на диаграмме вариантов использования интерфейс изображается в виде маленького круга, рядом с которым записывается его имя
- ✓ интерфейсы могут соединяться с вариантами использования сплошной линией со стрелкой/сплошной бз стрелки
- интерфейсы не могут содержать ни атрибутов, ни состояний, ни направленных ассоциаций

140 140 из 288

UML. Диаграмма классов

Для обозначение статических атрибутов и операций в UML используется

- ✓ Нижнее сплошное подчеркивание
- Нижнее пунктирное подчеркивание
- Верхнее сплошное подчеркивание
- Верхнее пунктирное подчеркивание

141 141 из 288

UML. Диаграмма классов

Для обозначения абстрактного класса его имя и абстрактные функции

- ✓ Используется курсив
- Используется жирный шрифт
- Используется подчёркивание
- всё перечисленное

142 142 из 288

UML. Диаграмма классов

Для обозначения отношения обобщения в языке UML используется

- [x] Сплошная линия с треугольной стрелкой на одном из концов. Стрелка указывает на более общий класс.
- [ ] Сплошная линия с треугольной стрелкой на одном из концов. Стрелка указывает на более производный класс.
- [ ] Пунктирная линия с треугольной стрелкой на одном из концов. Стрелка указывает на более общий класс.
- [ ] Пунктирная линия с треугольной стрелкой на одном из концов. Стрелка указывает на более производный класс.

143 143 из 288

UML. Диаграмма классов

Для обозначения реализации интерфейса в языке UML используется

- [x] слошная линия с кружком
- [ ] то значок "гнездо"
- [ ] пунктирная линия
- [ ] такой связи не существует

144 144 из 288

UML. Диаграмма классов

Зависимость – это…

- [x] такое отношение между классами, что изменение спецификации класса-поставщика может повлиять на работу зависимого класса, но не наоборот
- [ ] такое отношение между классами, что изменение спецификации класса-поставщика может повлиять на работу зависимого класса и наоборот
- [ ] такое отношение между интерфейсами, что изменение спецификации класса-поставщика может повлиять на работу зависимого класса и наоборот
- [ ] такое отношение между интерфейсами, что изменение спецификации класса-поставщика может повлиять на работу зависимого класса, но не наоборот

145 145 из 288

UML. Диаграмма классов

Постусловие операции это

- [x] условие, которое должно быть ложным, когда вызов операции успешно завершился, в предположении, что все предусловия были удовлетворены
- [ ] условие, которое должно быть ложным, когда вызов операции успешно завершился, в предположении, что не все предусловия были удовлетворены
- [ ] условие, которое должно быть ложным, когда вызов операции успешно завершился, в предположении, что все постусловия были удовлетворены
- [ ] условие, которое должно быть истиным, когда вызов операции успешно завершился, в предположении, что не все предусловия были удовлетворены

146 146 из 288

UML. Диаграмма классов

Самый слабый тип отношений между классами это

- [x] ассоциация
- [ ] агрегация
- [ ] композиция
- [ ] обобщение

147 147 из 288

Методы проектирования. SOLID

Антипаттерн – это

- [x] это распространённый подход к решению класса часто встречающихся проблем, являющийся неэффективным, рискованным или непродуктивным.
- [ ] это распространённый подход к решению класса часто встречающихся проблем, являющийся эффективным, безопасным и продуктивным
- [ ] это нераспространённый подход к решению класса часто встречающихся проблем, являющийся неэффективным, рискованным или непродуктивным.
- [ ] это один из принципов SOLID

148 148 из 288

Методы проектирования. SOLID

Выберите метод, являющийся методом рефакторинга

- [x] выделение класса
- [ ] замена полиморфизма условным оператором
- [ ] введение переменной
- [ ] отделение метода

149 149 из 288

Методы проектирования. SOLID

К принципам Grasp не относится

- [x] Наследование
- [ ] Полиморфизм
- [ ] Контроллер
- [ ] Создатель

150 150 из 288

Методы проектирования. SOLID

Принцип "The Open-Closed Principle" гласит

- [x] программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения
- [ ] модули верхних уровней не должны зависеть от модулей нижних уровней. Оба типа модулей должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.
- [ ] каждый объект должен иметь одну обязанность и эта обязанность должна быть полностью инкапсулирована в класс. Все его сервисы должны быть направлены исключительно на обеспечение этой обязанности.
- [ ] всё вышеперечисленное

151 151 из 288

Принцип "The Dependency Inversion Principle" гласит

- [ ] программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения
- [x] модули верхних уровней не должны зависеть от модулей нижних уровней. Оба типа модулей должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.
- [ ] каждый объект должен иметь одну обязанность и эта обязанность должна быть полностью инкапсулирована в класс. Все его сервисы должны быть направлены исключительно на обеспечение этой обязанности.
- [ ] всё вышеперечисленное

152 152 из 288

Методы проектирования. SOLID

Принцип "The Single Responsibility Principle" гласит

- программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения
- модули верхних уровней не должны зависеть от модулей нижних уровней. Оба типа модулей должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.
- ✓ каждый объект должен иметь одну обязанность и эта обязанность должна быть полностью инкапсулирована в класс. Все его сервисы должны быть направлены исключительно на обеспечение этой обязанности.
- клиенты не должны зависеть от методов, которые они не используют

153 153 из 288

Методы проектирования. SOLID

Принцип “The Liskov Substitution Principle” гласит

- ✓ наследующий класс должен дополнять, а не замещать поведение базового класса
- клиенты не должны зависеть от методов, которые они не используют
- программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения
- модули верхних уровней не должны зависеть от модулей нижних уровней. Оба типа модулей должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

154 154 из 288

Методы проектирования. SOLID

Принцип “The Interface Segregation Principle” гласит

- ✓ клиенты не должны зависеть от методов, которые они не используют
- наследующий класс должен дополнять, а не замещать поведение базового класса
- программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения
- каждый объект должен иметь одну обязанность и эта обязанность должна быть полностью инкапсулирована в класс. Все его сервисы должны быть направлены исключительно на обеспечение этой обязанности.

155 155 из 288

Методы проектирования. SOLID

Рефакторинг направлен на

- ✓ упрощения кода программы и сделать его более легким для понимания
- усложнения кода программы и сделать его более тяжёлым для понимания
- упрощения кода программы и сделать его более тяжёлым для понимания
- усложнения кода программы и сделать его более лёгким для понимания

156 156 из 288

Паттерны проектирования

К поведенческим паттернам относится паттерн

- [x] Снимок
- [ ] Абстрактная фабрика
- [ ] Фасад
- [ ] Легковес

157 157 из 288

Паттерны проектирования

К порождающим паттернам проектирования не относится

- [x] Всё перечисленное
- [ ] Стратегия
- [ ] Адаптер
- [ ] Посетитель

158 158 из 288

Паттерны проектирования

К структурным паттернам относится паттерн

- [x] Компоновщик
- [ ] Посетитель
- [ ] Состояние
- [ ] Шаблонный метод

159 159 из 288

Паттерны проектирования

Паттерн Абстрактная Фабрика

- [x] Порождающий паттерн проектирования, который позволяет создавать семейства связанных объектов, не привязываясь к конкретным классам создаваемых объектов.
- [ ] Структурный паттерн проектирования, который позволяет создавать семейства связанных объектов, не привязываясь к конкретным классам создаваемых объектов.
- [ ] Поведенческий паттерн проектирования, который позволяет создавать семейства связанных объектов, не привязываясь к конкретным классам создаваемых объектов.
- [ ] Структурный паттерн проектирования, который позволяет создавать семейства связанных объектов, привязываясь к конкретным классам создаваемых объектов.

160 160 из 288

Паттерны проектирования

Паттерн Адаптер

- [x] Структурный паттерн проектирования, который позволяет объектам с несовместимыми интерфейсами работать вместе.
- [ ] Поведенческий паттерн проектирования, который позволяет объектам с несовместимыми интерфейсами работать вместе.
- [ ] Порождающий паттерн проектирования, который позволяет объектам с несовместимыми интерфейсами работать вместе.
- [ ] Порождающий паттерн проектирования, который позволяет объектам с совместимыми интерфейсами работать вместе.

161 161 из 288

Паттерны проектирования

Паттерн Декоратор

- [x] Структурный паттерн проектирования, который позволяет динамически добавлять объектам новую функциональность, оборачивая их в полезные «обёртки».
- [ ] Поведенческий паттерн проектирования, который позволяет динамически добавлять объектам новую функциональность, оборачивая их в полезные «обёртки».
- [ ] Порождающий паттерн проектирования, который позволяет динамически добавлять объектам новую функциональность, оборачивая их в полезные «обёртки».
- [ ] Структурный паттерн проектирования, который позволяет динамически изменять объектам старую функциональность, оборачивая их в полезные «обёртки».

162 162 из 288

Паттерны проектирования

Паттерн Итератор

- [x] Поведенческий паттерн проектирования, который даёт возможность последовательно обходить элементы составных объектов, не раскрывая их внутреннего представления.
- [ ] Порождающий паттерн проектирования, который даёт возможность последовательно обходить элементы составных объектов, не раскрывая их внутреннего представления.
- [ ] Структурный паттерн проектирования, который даёт возможность последовательно обходить элементы составных объектов, не раскрывая их внутреннего представления.
- [ ] Структурный паттерн проектирования, который даёт возможность параллельно обходить элементы составных объектов, не раскрывая их внутреннего представления.

163 163 из 288

Паттерны проектирования

Паттерн Команда

- [x] Поведенческий паттерн проектирования, который превращает запросы в объекты, позволяя передавать их как аргументы при вызове методов, ставить запросы в очередь, логировать их, а также поддерживать отмену операций.
- [ ] Структурный паттерн проектирования, который превращает запросы в объекты, позволяя передавать их как аргументы при вызове методов, ставить запросы в очередь, логировать их, а также поддерживать отмену операций.
- [ ] Порождающий паттерн проектирования, который превращает запросы в объекты, позволяя передавать их как аргументы при вызове методов, ставить запросы в очередь, логировать их, а также поддерживать отмену операций.
- [ ] Поведенческий паттерн проектирования, который превращает запросы в объекты, позволяя передавать их как аргументы при вызове методов, ставить запросы в стек, логировать их, а также поддерживать отмену операций.

164 164 из 288

Паттерны проектирования

Паттерн Компоновщик

- [x] Структурный паттерн проектирования, который позволяет сгруппировать множество объектов в древовидную структуру, а затем работать с ней так, как будто это единичный объект.
- [ ] Порождающий паттерн проектирования, который позволяет сгруппировать множество объектов в древовидную структуру, а затем работать с ней так, как будто это единичный объект.
- [ ] Поведенческий паттерн проектирования, который позволяет сгруппировать множество объектов в древовидную структуру, а затем работать с ней так, как будто это единичный объект.
- [ ] Порождающий паттерн проектирования, который позволяет сгруппировать множество объектов в линейную структуру, а затем работать с ней так, как будто это единичный объект.

165 165 из 288

Паттерны проектирования

Паттерн Мост

- [x] Структурный паттерн проектирования, который разделяет один или несколько классов на две отдельные иерархии — абстракцию и реализацию, позволяя изменять их независимо друг от друга.
- [ ] Поведенческий паттерн проектирования, который разделяет один или несколько классов на две отдельные иерархии — абстракцию и реализацию, позволяя изменять их независимо друг от друга.
- [ ] Порождающий паттерн проектирования, который разделяет один или несколько классов на две отдельные иерархии — абстракцию и реализацию, позволяя изменять их независимо друг от друга.
- [ ] Структурный паттерн проектирования, который разделяет один или несколько классов на несколько отдельных иерархий — абстракцию, реализацию, агрегацию и другие, позволяя изменять их независимо друг от друга.

166 166 из 288

Паттерны проектирования

Паттерн Одиночка

- [x] Порождающий паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.
- [ ] Структурный паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.
- [ ] Поводенческий паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.
- [ ] Структурный паттерн проектирования, который гарантирует, что у класса есть несколько экземпляров, и предоставляет к нему глобальную точку доступа.

167 167 из 288

Паттерны проектирования

Паттерн Состояние

- [x] Поведенческий паттерн проектирования, который позволяет объектам менять поведение в зависимости от своего состояния. Извне создаётся впечатление, что изменился класс объекта.
- [ ] Структурный паттерн проектирования, который позволяет объектам менять поведение в зависимости от своего состояния. Извне создаётся впечатление, что изменился класс объекта.
- [ ] Порождающий паттерн проектирования, который позволяет объектам менять поведение в зависимости от своего состояния. Извне создаётся впечатление, что изменился класс объекта.
- [ ] все определения неверны

168 168 из 288

Паттерны проектирования

Паттерн Стратегия

- [x] Поведенческий паттерн проектирования, который определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс, после чего алгоритмы можно взаимозаменять прямо во время исполнения программы.
- [ ] Поведенческий паттерн проектирования, который определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс, после чего алгоритмы нельзя взаимозаменять прямо во время исполнения программы.
- [ ] Структурный паттерн проектирования, который определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс, после чего алгоритмы можно взаимозаменять прямо во время исполнения программы.
- [ ] Порождающий паттерн проектирования, который определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс, после чего алгоритмы можно взаимозаменять прямо во время исполнения программы.

169 169 из 288

Паттерны проектирования

Паттерн Строитель

- [x] Порождающий паттерн проектирования, который позволяет создавать сложные объекты пошагово. Строитель даёт возможность использовать один и тот же код строительства для получения разных представлений объектов.
- [ ] Поведенческий паттерн проектирования, который позволяет создавать сложные объекты пошагово. Строитель даёт возможность использовать один и тот же код строительства для получения разных представлений объектов.
- [ ] Структурный паттерн проектирования, который позволяет создавать сложные объекты пошагово. Строитель даёт возможность использовать один и тот же код строительства для получения разных представлений объектов.
- [ ] Структурный паттерн проектирования, который позволяет создавать простые объекты. Строитель даёт возможность использовать один и тот же код строительства для получения разных представлений объектов.

170 170 из 288

Паттерны проектирования

Паттерн Фабрика

- [x] Порождающий паттерн проектирования, который определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.
- [ ] Структурный паттерн проектирования, который определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.
- [ ] Поведенческий паттерн проектирования, который определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.
- [ ] Порождающий паттерн проектирования, который определяет частный интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.

171 171 из 288

Обобщённые типы

Выберите истинное высказывание относительно ограничений на обобщенные типы

- [ ] Можно создавать экземпляры параметров типа
- [x] Нельзя объявлять статические поля с типом параметра типа
- [ ] Возможно создавать массивы параметризованных типов
- [ ] Нельзя перегружать метод так, чтобы формальные параметры типа стирались в много типов и тот же сырой тип

172 172 из 288

Обобщённые типы

Выберите ложное высказывание относительно ограничений на обобщенные типы

- [x] Можно создавать экземпляры параметров типа
- [ ] Нельзя объявлять статические поля с типом параметра типа
- [ ] Невозможно создавать массивы параметризованных типов
- [ ] Нельзя перегружать метод так, чтобы формальные параметры типа стирались в один и тот же сырой тип

173 173 из 288

Обобщённые типы

Метасимвольный аргумент

- [x] Обозначается знаком ?
- [ ] Обозначается знаком !
- [ ] Обозначается знаком *
- [ ] Обозначается знаком ~

174 174 из 288

Обобщённые типы

Обобщенные типы задаются с помощью

- [x] С помощью буквы T в определении класса class Account<T>
- [ ] С помощью буквы T в параметре класса class Account(T)
- [ ] С помощью буквы T в определении класса class <T> Account
- [ ] нет правильного ответа

175 175 из 288

Обобщённые типы

Ограничение на метасимвольный аргумент, заданное с помощью ключевого слова extends

- [x] <? extends суперкласс >
- [ ] <? super подкласс >
- [ ] оба варинта правильны
- [ ] оба варинта неправильны

176 176 из 288

Обобщённые типы

Ограничение на метасимвольный аргумент, заданное с помощью ключевого слова super

- ○ <? extends суперкласс >
- ✓ <? super подкласс >
- ○ оба варинта правильны
- ○ оба варинта неправильны

177 177 из 288

Обобщённые типы

При использовании generic типов

- ✓ существует возможность создавать более статически типизированный код
- ○ существует возможность создавать более динамически типизированный код
- ○ существует возможность создавать более статически типизированную структуру
- ○ существует возможность создавать более динамически типизированную структуру

178 178 из 288

Обобщённые типы

При использовании non-generic типов

- ✓ теряется преимущество безопасности типов
- ○ не теряется преимущество безопасности типов
- ○ теряется преимущество опасности типов
- ○ generic знает только Иваненко (нет)

179 179 из 288

Коллекции

Базовая концепция Java — Collection — является

- ✓ базовым интерфейсом для всех коллекций и других интерфейсов коллекций
- ○ базовым интерфейсом не для всех коллекций и других интерфейсов коллекций
- ○ базовым интерфейсом только для коллекций
- ○ базовым интерфейсом только других интерфейсов коллекций

180 180 из 288

Коллекции

В какую иерархию коллекций входит Map

- [x] Представляет отдельную коллекцию
- [ ] Collection
- [ ] List
- [ ] Set

181 181 из 288

Коллекции

Выберите операцию, которая не является общей для разных типов коллекций

- [ ] add
- [ ] iterator
- [x] move
- [ ] remove

182 182 из 288

Коллекции

Итератор — это

- [x] объект, позволяющий просматривать содержимое коллекции последовательно, элемент за элементом
- [ ] объект, позволяющий просматривать содержимое коллекции параллельно
- [ ] объект, позволяющий просматривать содержимое коллекции с конца
- [ ] объект, позволяющий просматривать содержимое коллекции с середины

183 183 из 288

Коллекции

К коллекции, специально созданной для работы в многопоточном режиме, относится

- [x] Queue
- [ ] Stack
- [ ] List
- [ ] Map

184 184 из 288

Коллекции

Коллекция типа List — это

- [x] упорядоченная коллекция, в которой допустимы дублирующие значения и она представляет функциональность простых списков
- [ ] коллекция, состоящая из пар "ключ — значение"
- [ ] коллекция, предназначенная для хранения элементов в порядке, нужном для их обработки
- [ ] неупорядоченная коллекция, не содержащая повторяющихся элементов

185 185 из 288

Коллекции

Коллекция типа Map — это

- [ ] упорядоченная коллекция, в которой допустимы дублирующие значения и она представляет функциональность простых списков
- [x] коллекция, состоящая из пар "ключ — значение"
- [ ] коллекция, предназначенная для хранения элементов в порядке, нужном для их обработки
- [ ] неупорядоченная коллекция, не содержащая повторяющихся элементов

186 186 из 288

Коллекции

Коллекция типа Queue — это

- [ ] упорядоченная коллекция, в которой допустимы дублирующие значения и она представляет функциональность простых списков
- [ ] коллекция, состоящая из пар "ключ — значение"
- [x] коллекция, предназначенная для хранения элементов в порядке, нужном для их обработки
- [ ] неупорядоченная коллекция, не содержащая повторяющихся элементов

187 187 из 288

Коллекции

Коллекция типа Set — это

- [ ] упорядоченная коллекция, в которой допустимы дублирующие значения и она представляет функциональность простых списков
- [ ] коллекция, состоящая из пар "ключ — значение"
- [ ] коллекция, предназначенная для хранения элементов в порядке, нужном для их обработки
- [x] неупорядоченная коллекция, не содержащая повторяющихся элементов

188 188 из 288

Коллекции

Основные типы коллекций находятся в пакете

- [x] java.util
- [ ] java.lang
- [ ] java.base
- [ ] package

189 189 из 288

List и его типы

List в Java является

- [x] интерфейсом для операций с коллекцией, которая является списком
- [ ] интерфейсом для операций с коллекцией, которая является очередью
- [ ] интерфейсом для операций с коллекцией, которая является стеком
- [ ] интерфейсом для операций с коллекцией, которая является словарём

190 190 из 288

List и его типы

Интерфейс List не содержит описание следующего метода

- [ ] add
- [x] index
- [ ] remove
- [ ] set

191 191 из 288

List и его типы

К основным реализациям List не относится

- [ ] ArrayList
- [x] SortedList
- [ ] Vector
- [ ] Stack

192 192 из 288

List и его типы

Контейнер типа ArrayList представляет собой

- [x] Простой список объектов, т.е. инкапсулирует в себе обычный массив, длина которого автоматически увеличивается при добавлении новых элементов.
- [ ] Сложный список объектов, т.е. инкапсулирует в себе многомерный массив, длина которого автоматически увеличивается при добавлении новых элементов.
- [ ] Простой список объектов, т.е. инкапсулирует в себе обычный массив, длина которого автоматически увеличивается при добавлении старых элементов.
- [ ] нет правильных ответов

193 193 из 288

List и его типы

Контейнер типа LinkedList представляет собой

- [x] связанный список, т.е. структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и две ссылки на следующий и предыдущий узел списка.
- [ ] несвязанный список, т.е. структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и две ссылки на следующий и предыдущий узел списка.
- [ ] связанный список, т.е. структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну ссылку на следующий узел списка.
- [ ] связанный список, т.е. структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну ссылку на предыдущий узел списка.

194 194 из 288

List и его типы

Одна из особенностей ListIterator по сравнению с обычным Iterator в том

- [x] Iterator может использоваться для перебора элементов Set, List и Map. В отличие от него, ListIterator может быть использован только для перебора элементов коллекции List.
- [ ] Iterator позволяет перебирать элементы только в одном направлении, при помощи метода next(). Тогда как ArrayListIterator позволяет перебирать список в обоих направлениях, при помощи методов next() и previous().
- [ ] При помощи ListIterator вы можете модифицировать список, добавляя/удаляя/обновляя элементы с помощью методов add(), remove(), update(). Iterator не поддерживает данного функционала.
- [ ] Iterator позволяет перебирать элементы только в двух направлениях, при помощи метода next() и previous(). Тогда как ListIterator позволяет перебирать список в одном направлении, при помощи метода next().

195 195 из 288

Map и Set. Их подтипы

Map в Java является

- [x] Интерфейс Map<K, V> представляет отображение или иначе говоря словарь, где каждый элемент представляет пару "ключ-значение"
- [ ] Интерфейс Map<K> представляет отображение или иначе говоря словарь, где каждый элемент представляет пару "ключ"
- [ ] Интерфейс Map<V> представляет отображение или иначе говоря словарь, где каждый элемент представляет пару "значение"
- [ ] нет правильного варианта ответа (да, Ваня?)

196 196 из 288

Map и Set. Их подтипы

Взаимодействие потоков может осуществляться с помощью функции

- [ ] sleep()
- [ ] resume()
- [ ] stop()
- [x] join()

197 197 из 288

Map и Set. Их подтипы

Выберите ложное утверждение о многопоточном программировании

- [ ] порождённый процесс может состоять из нескольких потоков, выполняющихся «параллельно»
- [ ] позволяет достичь более эффективного использования ресурсов вычислительной машины
- [x] все потоки выполняются в адресном пространстве процесса и имеют общие дескрипторы классов
- [ ] выполняющийся процесс имеет как минимум один (главный) поток

198 198 из 288

Выберите ложное утверждение о множествах (Set)

- [ ] Интерфейс Set расширяет интерфейс Collection и представляет набор уникальных элементов.
- [ ] Set не добавляет новых методов, только вносит изменения унаследованные.
- [ ] Множества не содержат пары элементов e1 и e2, таких как e1.equals(e2), и не более одного null элемента.
- [x] нет ложных вариантов

199 199 из 288

Map и Set. Их подтипы

Выберите ложное утверждение об отображениях (Map)

- Интерфейс Map<K, V> представляет отображение или иначе говоря словарь, где каждый элемент представляет пару "ключ-значение"
- Все ключи уникальные в рамках объекта Map
- Map облегчает поиск элемента, если нам известен ключ - уникальный идентификатор объекта. Не расширяет интерфейс Collection
- [x] нет ложных вариантов

200 200 из 288

Map и Set. Их подтипы

Главная особенность реализаций SortedSet в

- [x] хранении элементов в отсортированном виде (сортировка по возрастанию)
- хранении элементов в отсортированном виде (сортировка по убыванию)
- хранении элементов в хаотичном виде
- Иваненко решит как хранить данные!

201 201 из 288

Map и Set. Их подтипы

Интерфейс Comparable предназначен для

- [x] упорядочивания объектов класса
- сравнения объектов класса
- добавления объектов класса
- удаления объектов класса

202 202 из 288

Map и Set. Их подтипы

Интерфейс Map не содержит описание следующего метода

- computeIfAbsent()
- getOrDefault()
- putIfAbsent()
- [x] removeAll()

203 203 из 288

Map и Set. Их подтипы

Итерация по отображению

- for (Map.Entry<String,String> entry : example.entrySet())
- for (String key : example.keySet())
- for (String value : example.values())
- Iterator<Map.Entry<String, String>> itr = example.entrySet().iterator(); while(itr.hasNext()) {
  Map.Entry<String, String> entry = itr.next();
  System.out.println("Key = " + entry.getKey() + ", Value = " + entry.getValue());
  }
- example.forEach((k, v) -> System.out.println("Key = " + k + ", Value = " + v));
- ✓ всё вышеперечисленное

204 204 из 288

Map и Set. Их подтипы

К основным реализациям Set не относится

- HashSet
- LinkedHashSet
- TreeSet
- ✓ SortedSet

205 205 из 288

Map и Set. Их подтипы

К отображению, которое сохраняет элементы в отсортированном порядке относится

- ✓ TreeMap
- LinkedHashSet
- HashSet
- SortedSet

206 206 из 288

Map и Set. Их подтипы

К преимуществам многопоточности не относится

- Улучшенная реакция приложения
- ✓ Тщательная разработка
- Эффективное использование ресурсов системы
- Более эффективное использование мультипроцессирования

207 207 из 288

Map и Set. Их подтипы

К состояниям потока не относится

- ○ выполнение
- ○ готовность
- ○ блокировка
- ✓ остановка

208 208 из 288

Map и Set. Их подтипы

Компараторы позволяют

- ✓ обеспечить полный контроль над порядком сортировки коллекций/массивов объектов
- ○ обеспечить неполный контроль над порядком сортировки коллекций/массивов объектов
- ○ обеспечить полный контроль над порядком сортировки только коллекций объектов
- ○ обеспечить полный контроль над порядком сортировки только массивов объектов

209 209 из 288

Map и Set. Их подтипы

Почему Map не расширяет интерфейс Collection?

- ✓ Collection не поддерживает пару "ключ-значение"
- ○ Collection поддерживает пару "ключ-значение"
- ○ Collection не хочет расширяться во имя Ивана Леонидовича
- ○ потому что (c) Слава Хаит (или Ваня Иваненко, как кому нравится)

210 210 из 288

Тестирование. Общие сведения

Test-Driven Development — это

- [x] техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг кода.
- [ ] техника разработки программного обеспечения, которая основывается на повторении очень длинных циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг кода.
- [ ] техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется код, который позволит пройти тест, затем пишется тест, покрывающий желаемое изменение, и под конец проводится рефакторинг кода.
- [ ] техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится ИНВЕНТАРИЗАЦИЯ.

211 211 из 288

Тестирование. Общие сведения

Библиотека JUnit применяется для

- [x] модульного тестирования программного обеспечения
- [ ] структурного тестирования программного обеспечения
- [ ] объектно-ориентированного тестирования программного обеспечения
- [ ] интеграционного тестирования программного обеспечения

212 212 из 288

Тестирование. Общие сведения

Выберите истинное утверждение

- [x] Библиотека JUnit применяется для модульного тестирования программного обеспечения
- [ ] Библиотека JUnit применяется для структурного тестирования программного обеспечения
- [ ] Исчерпывающее тестирование (входное или тестирование путей) это методика, в которой набор тестов включает в себя все комбинации выходных данных и предусловий
- [ ] Исчерпывающее тестирование (входное или тестирование путей) это методика, в которой набор тестов включает в себя все комбинации входных данных и постдусловий

213 213 из 288

Тестирование. Общие сведения

Выберите ложное утверждение

- [x] Библиотека JUnit применяется для структурного тестирования программного обеспечения
- [ ] Библиотека JUnit применяется для модульного тестирования программного обеспечения
- [ ] Не существует стратегии тестирования методом «красного ящика» (red-box testing). Исчерпывающее выходное тестирование
- [ ] Тестирование – это процесс или последовательность процессов, позволяющих удостовериться в том, что программный код делает все то, для чего он предназначался, и наоборот, не делает того, для чего не предназначался.

214 214 из 288

Тестирование. Общие сведения

Исчерпывающее тестирование (входное или тестирование путей)

- [x] это методика, в которой набор тестов включает в себя все комбинации входных данных и предусловий
- [ ] это методика, в которой набор тестов включает в себя все комбинации выходных данных и предусловий
- [ ] это методика, в которой набор тестов включает в себя все комбинации входных данных и постдусловий
- [ ] это методика, в которой набор тестов включает в себя все комбинации выходных данных и постдусловий

215 215 из 288

Тестирование. Общие сведения

Не существует стратегии тестирования методом

- [ ] Тестирование методом «черного ящика» (black-box testing, data-driven testing, input/output-driven testing). Исчерпывающее входное тестирование
- [ ] Тестирование методом «белого ящика» (white-box testing). Исчерпывающее тестирование путей
- [x] Тестирование методом «красного ящика» (red-box testing). Исчерпывающее выходное тестирование
- [ ] Сочетание методологий – тестирование методом «серого» ящика (grey-box testing).

216 216 из 288

Тестирование. Общие сведения

Тестирование – это

- [x] процесс или последовательность процессов, позволяющих удостовериться в том, что программный код делает все то, для чего он предназначался, и наоборот, не делает того, для чего не предназначался.
- [ ] процесс или последовательность процессов, позволяющих удостовериться в том, что программный код делает не все то, для чего он предназначался, и наоборот, делает то, для чего не предназначался.
- [ ] процесс или последовательность процессов, позволяющих удостовериться в том, что программный код делает все то, для чего он предназначался, и делает того, для чего не предназначался.
- [ ] процесс или последовательность процессов, позволяющих удостовериться в том, что программный код не делает все то, для чего он предназначался, делает то, для чего не предназначался.

217 217 из 288

Тестирование. Общие сведения

Тестирование методом белого ящика предполагает, что

- [x] он базируется только лишь на коде программы
- [ ] он только лишь на тестировании по функциональной спецификации и требованиям, при этом не смотря во внутреннюю структуру кода и без доступа к базе данных
- [ ] оба варианта верны
- [ ] оба варианта неверны

218 218 из 288

Тестирование. Общие сведения

Тестирование методом черного ящика предполагает, что

- [ ] он базируется только лишь на коде программы
- [x] он только лишь на тестировании по функциональной спецификации и требованиям, при этом не смотря во внутреннюю структуру кода и без доступа к базе данных
- [ ] оба варианта верны
- [ ] оба варианта неверны

219 219 из 288

Тестирование. Методы тестирования

К методам тестирования "белого ящика" не относятся

- [x] Прогнозирование ошибок
- [ ] Комбинаторное покрытие условий
- [ ] Покрытие решений и условий
- [ ] Покрытие операторов – добиться выполнения каждой инструкции программы

220 220 из 288

Тестирование. Методы тестирования

К методам тестирования "черного ящика" не относится

- [x] Комбинаторное покрытие условий
- [ ] Эквивалентное разбиение
- [ ] Прогнозирование ошибок
- [ ] Анализ граничных значений

221 221 из 288

Тестирование. Методы тестирования

Тестирование методом анализа граничных значений

- [x] тестовые сценарии должны давать возможность программе функционировать на границах и со значениями внутри и вне границ.
- [ ] при проверке диапазона значений после выбора набора данных, находящихся в недопустимых пределах, следует проверить, как программа ведет себя с граничными значениями допустимых пределов
- [ ] анализ граничных значений редко используется при проверке диапазона чисел
- [ ] для каждого диапазона не существует границ

222 222 из 288

Тестирование. Методы тестирования

Тестирование методом комбинаторного покрытия условий

- [x] требуют написание тестов, которые реализуют все возможные комбинации истинности условий в каждом решении.
- [ ] требуют написание тестов, которые реализуют все возможные комбинации ложности условий в каждом решении.
- [ ] требуют написание тестов, которые реализуют все возможные комбинации истинности условий в одном решении.
- [ ] требуют написание тестов, которые реализуют все возможные комбинации ложности условий в одном решении.

223 223 из 288

Тестирование. Методы тестирования

Тестирование методом покрытия операторов

- [x] требует написания такого количества тестов, чтобы при выполнении их всех каждый оператор был выполнен хотя бы один раз.
- [ ] требует написания такого количества тестов, чтобы при выполнении их всех каждый оператор был выполнен несколько раз.
- [ ] требует написания одного теста, чтобы при выполнении его каждый оператор был выполнен хотя бы один раз.
- [ ] требует написания одного теста, чтобы при выполнении его каждый оператор был выполнен несколько раз.

224 224 из 288

Тестирование. Методы тестирования

Тестирование методом покрытия решений

- под решениями здесь подразумеваются высказывания в операторах условного перехода, выбора, цикла, взятые целиком.
- эти высказывания часто составлены из более простых – элементарных высказываний.
- метод покрытия решений требует такого количества тестов, чтобы при выполнении их всех по каждой траектории, соединяющей соседние элементы блок-схемы вычисление прошло хотя бы один раз.
- ✓ все варианты верны

225 225 из 288

Тестирование. Методы тестирования

Тестирование методом покрытия условий

- ✓ состоит в таком подборе тестов, когда каждое условие (элементарное суждения в условных операторах) принимает как истинное так и ложное значение.
- состоит в таком подборе тестов, когда каждое условие (элементарное суждения в условных операторах) принимает истинное значение.
- состоит в таком подборе тестов, когда каждое условие (элементарное суждения в условных операторах) принимает ложное значение.
- состоит в таком подборе тестов, когда каждое условие (элементарное суждения в условных операторах) не принимает никакого значения

226 226 из 288

Тестирование. Методы тестирования

Тестирование методом эквивалентного разбиения

- ✓ - исключить набор входных данных, которые заставляют систему вести себя одинаково и давать одинаковый результат при тестировании программы.
- - исключить набор выходных данных, которые заставляют систему вести себя одинаково и давать одинаковый результат при тестировании программы.
- - исключить набор входных данных, которые заставляют систему вести себя одинаково и давать разный результат при тестировании программы.
- - исключить набор выходных данных, которые заставляют систему вести себя одинаково и давать разный результат при тестировании программы.

227 227 из 288

Тестирование. Методы тестирования

Причинно-следственные диаграммы

- [x] графическое изображение, которое в сжатой форме и логической последовательности распределяет причины.
- [ ] графическое изображение, которое в векторной форме и логической последовательности распределяет причины.
- [ ] графическое изображение, которое в сжатой форме и физической последовательности распределяет причины.
- [ ] графическое изображение, которое в векторной форме и физической последовательности распределяет причины.

228 228 из 288

Тестирование. Его методы

Инкрементное тестирование – это

- [x] тестирование, где модули не тестируются изолированно друг от друга, а постепенно подключаются к набору уже протестированных модулей и тестируются в составе сборки.
- [ ] тестирование, где модули тестируются изолированно друг от друга, а постепенно подключаются к набору уже протестированных модулей и тестируются в составе сборки.
- [ ] тестирование, где модули не тестируются изолированно друг от друга, а сразу подключаются к набору уже протестированных модулей и тестируются в составе сборки.
- [ ] тестирование, где модули тестируются изолированно друг от друга и сразу подключаются к набору уже протестированных модулей и тестируются в составе сборки.

229 229 из 288

Тестирование. Его методы

К достоинствам восходящего тестирования относится

- [ ] Имеет преимущества, если основные ошибки встречаются главным образом на нижних иерархических уровнях программы
- [ ] Легче создавать тестовые условия
- [ ] Проще обеспечить наблюдение за результатами тестирования
- [x] Всё вышеперечисленное

230 230 из 288

Тестирование. Его методы

К достоинствам нисходящего тестирования относится

- [ ] Имеет преимущества, если основные ошибки встречаются главным образом на верхних иерархических уровнях программы
- [ ] Представление тестов упрощается после подключения функций ввода-вывода
- [ ] Ранее формирование каркаса программы обеспечивает возможность демонстрации ее работы
- [x] Всё вышеперечисленное

231 231 из 288

Тестирование. Его методы

К недостаткам восходящего тестирования относится

- Необходимо разрабатывать модули-драйверы
- Программа как единое целое не существует до тех пор, пока не добавлен последний модуль
- Оба варианта верны
- Оба варианта - ложь

232 232 из 288

Тестирование. Его методы

К недостаткам нисходящего тестирования относится

- Необязательно разрабатывать модули-заглушки
- Модули-заглушки часто оказываются сложнее, чем предполагалось изначально
- До подключения функций ввода-вывода представление тестовых данных в заглушках может вызывать затруднения
- Невозможность или значительная сложность создания тестовых условий

233 233 из 288

Тестирование. Его методы

Модульное тестирование – это

- процесс тестирования отдельных блоков, подпрограмм, классов и процедур, образующих крупную программу.
- процесс тестирования маленькой программы
- процесс компиляции отдельных блоков, подпрограмм, классов и процедур, образующих крупную программу.
- процесс тестирования всей программы целиком, без деления

234 234 из 288

Тестирование. Его методы

Неинкрементное тестирование – это

- Сначала выполняется модульное тестирование всех модулей, причем каждый тестируется как независимая сущность.
- Модули объединяются или интегрируются в программу
- Модули могут тестироваться параллельно.
- Модули могут тестироваться последовательно.

235 235 из 288

Тестирование. Категории и цели

Категория "Возможности" в системном тесте предполагает

- [x] Проверяется полнота реализации функциональных возможностей, определенных целями
- [ ] Тестируется способность средств восстановления выполнять свои функции
- [ ] Определяется соответствие программы специфицированным показателям надежности, таким как длительность непрерывной работы и среднее время наработки на отказ
- [ ] неверно всё перечисленное

236 236 из 288

Тестирование. Категории и цели

Категория "Восстанавливаемости" в системном тесте предполагает

- [ ] Проверяется полнота реализации функциональных возможностей, определенных целями
- [x] Тестируется способность средств восстановления выполнять свои функции
- [ ] Определяется соответствие программы специфицированным показателям надежности, таким как длительность непрерывной работы и среднее время наработки на отказ
- [ ] неверно всё перечисленное

237 237 из 288

Тестирование. Категории и цели

Категория "Надежности" в системном тесте предполагает

- [ ] Проверяется полнота реализации функциональных возможностей, определенных целями
- [ ] Тестируется способность средств восстановления выполнять свои функции
- [x] Определяется соответствие программы специфицированным показателям надежности, таким как длительность непрерывной работы и среднее время наработки на отказ
- [ ] неверно всё перечисленное

238 238 из 288

Тестирование. Категории и цели

Цель интеграционного теста

- [x] Тестирование модулей программы, объединенных в группу. Проходит после модульного.
- [ ] Сравнение функций, реализуемых модулем, со спецификациями, описывающими его функциональные или интерфейсные характеристики.
- [ ] Формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:
  - определения удовлетворяет ли система приемочным критериям;
  - вынесения решения заказчиком или другим уполномоченным лицом принимается приложение или нет
- [ ] Проверка функциональных и нефункциональных требований системы в целом, не принимая во внимания ее отдельные компоненты.

239 239 из 288

Тестирование. Категории и цели

Цель модульного теста

- [x] Сравнение функций, реализуемых модулем, со спецификациями, описывающими его функциональные или интерфейсные характеристики.
- [ ] Формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:
  - определения удовлетворяет ли система приемочным критериям;
  - вынесения решения заказчиком или другим уполномоченным лицом принимается приложение или нет
- [ ] Проверка функциональных и нефункциональных требований системы в целом, не принимая во внимания ее отдельные компоненты.
- [ ] Тестирование установки направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.

240 240 из 288

Тестирование. Категории и цели

Цель приемочного теста

- [x] Формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:
  - определения удовлетворяет ли система приемочным критериям;
  - вынесения решения заказчиком или другим уполномоченным лицом принимается приложение или нет
- [ ] Проверка функциональных и нефункциональных требований системы в целом, не принимая во внимания ее отдельные компоненты.
- [ ] Тестирование установки направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.
- [ ] Процесс, нацеленный на выявление расхождений между поведением программы и внешней спецификацией.

241 241 из 288

Тестирование. Категории и цели

Цель системного теста

- [x] Проверка функциональных и нефункциональных требований системы в целом, не принимая во внимания ее отдельные компоненты.
- [ ] Тестирование установки направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.
- [ ] Процесс, нацеленный на выявление расхождений между поведением программы и внешней спецификацией.
- [ ] Тестирование модулей программы, объединенных в группу. Проходит после модульного.

242 242 из 288

Тестирование. Категории и цели

Цель тестирования установки

- [x] Тестирование установки направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.
- [ ] Процесс, нацеленный на выявление расхождений между поведением программы и внешней спецификацией.
- [ ] Тестирование модулей программы, объединенных в группу. Проходит после модульного.
- [ ] Формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:
  - определения удовлетворяет ли система приемочным критериям;
  - вынесения решения заказчиком или другим уполномоченным лицом принимается приложение или нет

243 243 из 288

Тестирование. Категории и цели

Цель функционального теста

- [x] Процесс, нацеленный на выявление расхождений между поведением программы и внешней спецификацией.
- [ ] Тестирование модулей программы, объединенных в группу. Проходит после модульного.
- [ ] Проверка функциональных и нефункциональных требований системы в целом, не принимая во внимания ее отдельные компоненты.
- [ ] Формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:
  - определения удовлетворяет ли система приемочным критериям;
  - вынесения решения заказчиком или другим уполномоченным лицом принимается приложение или нет

244 244 из 288

Тестирование. Категории и цели

К критериям завершения тестов не относится

- ○ Время
- ○ Бюджет
- ○ Все тест-кейсы пройдены, найденные баги исправлены и перепроверены
- ✓ Всё вышеперечисленное

245 245 из 288

JavaFX

Объект, представляющий собой окно программы и содержащий все объекты JavaFX-приложения называется

- ✓ Stage
- ○ Scene
- ○ FXMLLoader
- ○ Application

246 246 из 288

JavaFX

Объект, представляющий физический контент JavaFX-приложения называется

- ✓ Scene
- ○ Stage
- ○ FXMLLoader
- ○ Application

247 247 из 288

JavaFX

Основным паттерном, используемым при разработке JavaFX-приложения, является

- ✓ MVC
- ○ MVP
- ○ MVVP
- ○ MVI

248 248 из 288

JavaFX

Разметка JavaFX-приложения хранится в файле с расширением

- [x] fxml
- [ ] xml
- [ ] javaxml
- [ ] fxxml

249 249 из 288

JavaFX

Макет JavaFX, размещающий все компоненты приложения последовательно друг на друге, называется

- [x] StackPane
- [ ] HBox
- [ ] GridPane
- [ ] AnchorPane

250 250 из 288

JavaFX

Макет JavaFX, добавляющий компоненты приложения в форме плиток одинакового размера, называется

- [x] GridPane
- [ ] AnchorPane
- [ ] StackPane
- [ ] HBox

251 251 из 288

JavaFX

Одновременные манипуляции, выполняемые несколькими потоками, над графом сцены JavaFX

- [x] недопустимы
- [ ] допустимы
- [ ] не знаю
- [ ] Я Иван Леонидович Иваненко. Где правильный ответ

252 252 из 288

JavaFX

Классы, применяемые для организации многопоточности в JavaFX, называются

- [x] Task и Worker
- [ ] Task и Application
- [ ] Task и DataOutputStream
- [ ] Task и Socket

253 253 из 288

SQL. JDBC

Этот интерфейс применяется при использовании статических SQL-запросов, не изменяющихся в процессе работы

- [x] Statement
- [ ] PreparedStatement
- [ ] ResultSet
- [ ] PreparedResultSet

254 254 из 288

SQL. JDBC

Этот интерфейс применяется, если SQL-запросы используют параметры, которые многократно изменяются в процессе работы

- [ ] Statement
- [x] PreparedStatement
- [ ] ResultSet
- [ ] PreparedResultSet

255 255 из 288

SQL. JDBC

Метод Statement, возвращающий ResultSet-объект, называется

- [x] ResultSet executeQuery (String SQL)
- [ ] Statement executeQuery (String SQL)
- [ ] PreparedStatement executeQuery (String SQL)
- [ ] PreparedResultSet executeQuery (String SQL)

256 256 из 288

SQL. JDBC

Интерфейс CallableStatement применяется для

- [x] организации доступа к хранимым процедурам БД
- [ ] организации доступа к хранимым процедурам файлах
- [ ] организации доступа к хранимым процедурам облака
- [ ] организации доступа к хранимым процедурам хеш-данных

257 257 из 288

SQL. JDBC

С помощью этого интерфейса можно получить доступ к данным, полученным оператором SELECT

- [ ] Statement
- [ ] PreparedStatement
- [x] ResultSet
- [ ] PreparedResultSet

258 258 из 288

SQL. JDBC

По выборке, получаемой с помощью объекта ResultSet, можно двигаться

- [ ] ResultSet.TYPE_FORWARD_ONLY
- [ ] ResultSet.TYPE_SCROLL_INSENSITIVE
- [ ] ResultSet.TYPE_SCROLL_SENSITIVE
- [x] всё перечисленное

259 259 из 288

SQL. JDBC

Выборка, получаемая с помощью ResultSet является

- [x] по умолчанию read_only
- [ ] по умолчанию write_only
- [ ] по умолчанию update_only
- [ ] по умолчанию delete_only

260 260 из 288

SQL. JDBC

Ключевая особенность транзакций -

- [x] либо выполняется полностью, либо не выполняется вообще
- [ ] либо выполняется полностью
- [ ] не выполняется вообще
- [ ] либо не выполняется полностью, либо выполняется вообще

261 261 из 288

SQL. JDBC

Типы данных JDBC

- [ ] BLOB
- [ ] ARRAY
- [ ] TIME
- [x] всё перечисленное

262 262 из 288

SQL. JDBC

Для закрепления изменений, произведенных транзакцией

- [x] conn.commit();
- [ ] conn.push();
- [ ] conn.pull();
- [ ] conn.merge();

263 263 из 288

SQL. JDBC

Для включения режима транзакций при наличии объекта Connection conn используется

- [x] conn.setAutoCommit(false)
- [ ] conn.setAutoCommit(true)
- [ ] conn.getAutoCommit(true)
- [ ] conn.getAutoCommit(false)

264 264 из 288

SQL. JDBC

Для внесения изменений в таблицу базы данных

- [x] updateRow()
- [ ] addRow()
- [ ] cancelRow()
- [ ] всё перечисленное

265 265 из 288

Функциональный подход. Потоковые операции

При использовании ссылок на методы можно ссылаться на

- [ ] Константный метод
- [ ] Параметризованный массив
- [ ] Метод объекта
- [x] Конструктор

266 266 из 288

Функциональный подход. Потоковые операции

Функция является чистой, если

- [ ] Выполнение функции не имеет побочных эффектов
- [ ] Возвращаемое функцией значение зависит только от входных параметров
- [x] оба варианта верны
- [ ] оба варианта - ложь

267 267 из 288

Функциональный подход. Потоковые операции

Функция имеет высший порядок, если

- [ ] Функция принимает один или более функций в качестве параметров
- [ ] Функция возвращает другую функцию в качестве результата
- [x] оба варианта верны
- [ ] оба варианта - ложь

268 268 из 288

Функциональный подход. Потоковые операции

Функция не изменяет состояние, если

- [x] Идея неизменяемости заключается в том, что созданное значение никогда не может быть изменено
- [ ] Идея изменяемости заключается в том, что созданное значение никогда не может быть изменено
- [ ] Идея неизменяемости заключается в том, что созданное значение может быть изменено
- [ ] Идея изменяемости заключается в том, что созданное значение может быть изменено

269 269 из 288

Функциональный подход. Потоковые операции

Функциональный интерфейс — это

- [x] интерфейс, который содержит лишь один абстрактный метод.
- [ ] интерфейс, который содержит несколько абстрактный метод.
- [ ] интерфейс, который не содержит абстрактный метод.
- [ ] нет правильных ответов

270 270 из 288

Функциональный подход. Потоковые операции

Для сопоставления лямбда-выражения и интерфейса необходимо

- [ ] Интерфейс содержит только один абстрактный (нереализованный) метод
- [ ] Параметры лямбда-выражения совпадают с параметрами единственного метода
- [ ] Тип возвращаемого значения лямбда-выражения совпадает с типом возвращаемого значения единственного метода функционального интерфейса
- [x] всё перечисленное

271 271 из 288

Функциональный подход. Потоковые операции

К встроенным функциональным интерфейсам Java не относится

- [ ] UnaryOperator
- [ ] BinaryOperator
- [ ] Function
- [x] Operator

272 272 из 288

Функциональный подход. Потоковые операции

Функциональный интерфейс, представляющий функцию, принимающий один параметр и возвращающий единственное значение, это (выбрать наиболее подходящий)

- [x] Function
- [ ] Predicate
- [ ] UnaryOperator
- [ ] Consumer

273 273 из 288

Функциональный подход. Потоковые операции

Функциональный интерфейс, представляющий функцию, принимающий один параметр и возвращающий булево значение, это (выбрать наиболее подходящий)

- [ ] Function
- [x] Predicate
- [ ] UnaryOperator
- [ ] Consumer

274 274 из 288

Функциональный подход. Потоковые операции

Функциональный интерфейс, представляющий функцию, принимающий один параметр и возвращающий значение того же типа, это (выбрать наиболее подходящий)

- [ ] Function
- [ ] Predicate
- [x] UnaryOperator
- [ ] Consumer

275 275 из 288

Функциональный подход. Потоковые операции

Consumer — это функциональный интерфейс, представляющий функцию, которая

- [x] принимает один параметр без возврата значения.
- [ ] принимает несколько параметров без возврата значения.
- [ ] принимает один параметр с возвратом значения.
- [ ] не принимает параметры без возврата значения.

276 276 из 288

Функциональный подход. Потоковые операции

Техника, которая комбинирует множественные функции в единую функцию, которая использует все комбинируемые функции, это

- [x] Функциональная композиция
- [ ] Структурная композиция
- [ ] Объектно-ориентированная композиция
- [ ] Протокольная композиция

277 277 из 288

Функциональный подход. Потоковые операции

Функциональная композиция предикатов может выполняться

- [x] функциями and() и or()
- [ ] функция and()
- [ ] функция or()
- [ ] функция имени Александры Александровны Нероды

278 278 из 288

Функциональный подход. Потоковые операции

Функциональная композиция функций может выполняться

- [x] функциями compose() и andThen()
- [ ] функцией compose()
- [ ] функцией andThen()
- [ ] функция имени Анастасии Петровны Воробей

279 279 из 288

Функциональный подход. Потоковые операции

Что такое Java Stream API

- [x] JSA предоставляет функциональный подход к обработке коллекции объектов.
- [ ] JSA предоставляет структурный подход к обработке коллекции объектов.
- [ ] JSA предоставляет объектно-ориентированный подход к обработке коллекции объектов.
- [ ] JSA предоставляет протокольный подход к обработке коллекции объектов.

280 280 из 288

Функциональный подход. Потоковые операции

Нетерминальная потоковая операция это

- [x] возвращает трансформированный поток
- [ ] возвращает нетрансформированный поток
- [ ] ничего не возвращает
- [ ] возвращает радость Алексея Сергеевича Луда

281 281 из 288

Функциональный подход. Потоковые операции

Терминальная потоковая операция это

- [x] возвращает конкретный результат
- [ ] возвращает расплывчатый результат
- [ ] не возвращает результат
- [ ] возвращает радость Павла Иннокентьевича Галанина

282 282 из 288

Функциональный подход. Потоковые операции

К терминальным потоковым операциям не относится метод

- [ ] findFirst
- [ ] findAny
- [ ] allMatch
- [x] match

283 283 из 288

Функциональный подход. Потоковые операции

К терминальным потоковым операциям относится метод

- [ ] filter
- [ ] distinct
- [x] count
- [ ] flatMap

284 284 из 288

Функциональный подход. Потоковые операции

К нетерминальным потоковым операциям не относится метод

- ○ filter
- ○ map
- ○ skip
- ✓ count

285 285 из 288

Функциональный подход. Потоковые операции

К нетерминальным потоковым операциям относится метод

- ✓ limit
- ○ reduce
- ○ collect
- ○ count

286 286 из 288

Функциональный подход. Потоковые операции

!!!К недостаткам Java Stream API относится!!!

- ○ неизменяемые данные
- ○ независимые данные
- ○ проверка исключений
- ✓ всё перечисленное

287 287 из 288

Функциональный подход. Потоковые операции

Java Stream-объект может быть создан

- ✓ из коллекции
- ○ из массива
- ○ из стека
- ○ из очереди

288 288 из 288

Функциональный подход. Потоковые операции

!!!Редукция Java Stream-объекта применяется!!!

- [x] для терминальных операций сведения, возвращая некоторое значение - результат операции
- [ ] для терминальных операций сведения, возвращая некоторое значение - переменную
- [ ] для терминальных операций сведения, ничего не возвращая
- [ ] для терминальных операций сведения, возвращая некоторое значение - Станислава Николаевича Коташевича