**SOLID** - это принципы разработки программного обеспечения, следуя которым Вы получите хороший код, который в дальнейшем будет хорошо масштабироваться и поддерживаться в рабочем состоянии.

**S - Single Responsibility Principle - принцип единственной ответственности**. Каждый класс должен иметь только одну зону ответственности.

**O - Open closed Principle - принцип открытости-закрытости.** Классы должны быть открыты для расширения, но закрыты для изменения.

**L - Liskov substitution Principle - принцип подстановки Барбары Лисков.** Должна быть возможность вместо базового (родительского) типа (класса) подставить любой его подтип (класс-наследник), при этом работа программы не должна измениться.

**I -  Interface Segregation Principle - принцип разделения интерфейсов.** Данный принцип обозначает, что не нужно заставлять клиента (класс) реализовывать интерфейс, который не имеет к нему отношения.

**D - Dependency Inversion Principle - принцип инверсии зависимостей.** Модули верхнего уровня не должны зависеть от модулей нижнего уровня. И те, и другие должны зависеть от абстракции. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

**1. Напишите команду с помощью которой можно создать файл с указанием файлами?**

(открытый)

manifest.mf

Name.class

!!jar -cm manifest.mf Name.class!!

**2. Для чего нужна команда jdb? (с вариантами ответов)**

!!для отладки!!

**3. Как правильно объявить массив (с несколькими вариантами ответов)**

int x = new int[2];

!!int x[] = new int[2];!!

!!int[] x = new int[2];!!

int[] x[] = new int[2];

**4. Что выведет программа в результате выполнения?**

int x = 14;

x-= x-- + --x;

print(x) !!-12!!

**5. Что выведет программа в результате выполнения?**

for (i=0; i<=10; i+=6)

print(i--); !!0510!!

**6. Что выведет программа в результате выполнения?**

class HelloWorld {

public static int m() {

int x = 10;

}

public static void main(String[] args) {

int z = m();

}

}

!!Ошибка компиляции!!

**7. Какое максимальное значение может принимать переменная var?**

byte[] x = new byte[var]; !!Integer.MAX\_VALUE, 2147483647, 2\*\*31-1!!

**8. Каковы особенности использования this в статических блоках инициализации?**

!!их нельзя там использовать!!

**9. Напишите конструктор по умолчанию, который инициализирует поля класса? (вопрос под**

**вопросом)**

!!public Anumal() { } !!Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

SOLID:

**Какие из принципов SOLID нарушены в коде?**

public interface Animal {

void run();

void jamp();

}

• S

• O

• L

• !!I!!

• D

**2. Какие из принципов SOLID нарушены в коде?**

public class Car {

private Wheel[] wheels;

public Car(Wheel[] wheels) {

this.wheels =wheels;

}

}

• S

• O

• L

• I

• !!D!!

**3. Какие из принципов SOLID нарушены в коде?**

public class Loser {

public int age;

public int luck;

}

• !!S!!

• !!O!!

• L

• I

• D

**4. Укажите все переопределённые методы**

public class MyClass {

@Override

public int hashCode()

{ return super.hashCode(); }

public String toString()

{ return “MyClass {}”; }

@Override

public int clone() { return null; }

}

• equals

• save

• !!hashCode!!

• asString

• Другое: !!toString(), clone()!!

**5. Вызвать метод doit**

class B {

void doit(){}

class A {

void randomName(){

//вызовите doit

}

}

}

• !!doit(); или B.this.doit();!!

**6. Укажите максимально абстрактный допустимый тип данных X, при котором этот код**

**скомпилируется**

class Thrower {

public static void go(X x) throws X {

throw x;

}

}

• !!Throwable!!

**7. Укажите все свойства checked исключений**

• Блок finally обязательный (block finally required)

• !!Генерируются компилятором (generated by compiler)!!

• Генерируются JVM (generated by JVM)

• Без try-catch не скомпилируется (won't compile without try-catch)

• !!try-catch или throws обязательны (try-catch or throws are required)!!

• !!Наследуются от Throwable (inherited from Throwable)!!

• Наследуются от Error (inherited from Throwable)

• Не могут иметь конструктора по умолчанию (can't have default constructor)

• Не содержат в себе стек вызовов (don't contains stacktrace)

**8. Что есть в enum по дефолту?**

• !!name()!!

• !!toString()!!

• applySelfEffect()

• order()

• !!hashCode()!!

• sayPhrase()

• !!завалитьРубежку()!!

**9. Укажите все свойства рекордов**

• !!нельзя наследовать от record, не имеет других полей (кроме конструктора)!!

**10. Что нужно исправить чтобы код скомпилился?**

interface Car {

void Print() {System.out.println(“ПСЖ”);}

}

• public interface => private interface

• interface => default interface

• interface => functional interface??

• !!public void => default void!!

• нет ошибок