Тест по уроку «Принципы программирования»

Gesamtpunktzahl 14/18

✓ Какой тип данных в Java подходит для хранения строкового текста? * 1/1

- A. int
- B. double
- C. String ✓
- D. boolean

✓ Какая управляющая структура выполняет блок кода до тех пор, пока условие истинно? 1/1

- A. if
- B. switch
- C. for
- D. while ✓

Feedback

Объяснение: while проверяет условие перед каждой итерацией и повторяет блок, пока условие остаётся истинным; for обычно используется для счётных циклов, а if и switch — для ветвления.

1/1

Что позволяет организовать код в логические блоки, принимать параметры и возвращать результаты?

- A. Переменные
- B. Классы
- C. Циклы
- D. Функции и методы ✓

1/1

Согласно принципам чистого кода, комментарии должны использоваться для:

- A. Подробного описания того, что делает код
- B. Объяснения причин, по которым принято то или иное решение ✓
- C. Копирования и вставки фрагментов кода
- D. Автоматизации сборки проекта

1/1

Что означает принцип YAGNI (You Aren’t Gonna Need It)?

- A. Не дублировать код
- B. Избегать избыточных зависимостей
- C. Не добавлять функциональность до тех пор, пока она действительно не понадобится ✓
- D. Делать код максимально документируемым

✓ Что означает принцип DRY (Don’t Repeat Yourself)? 1/1

- A. Создавать классы с одной ответственностью
- B. Избегать дублирования кода ✓
- C. Делать методы короче пяти строк
- D. Использовать как можно меньше комментариев

✗ Какой принцип SOLID утверждает, что класс должен иметь только одну причину для изменения? 0/1

- A. OCP
- B. ISP
- C. SRP
- D. DIP ✗

Richtige Antwort

- C. SRP

Feedback

Объяснение: SRP требует, чтобы класс выполнял лишь одну единственную задачу, обеспечивая одну ответственность.

✓ 1/1

Какой принцип SOLID гласит, что программные сущности должны быть открыты для расширения, но закрыты для модификации?

- A. LSP
- B. OCP ✓
- C. ISP
- D. SRP

✓ 1/1

Почему следующий пример нарушает Liskov Substitution Principle (LSP)?

```
class Bird {
    void fly() { … }
}
class Penguin extends Bird {
    @Override
    void fly() { throw new UnsupportedOperationException(); }
}
```

- A. Потому что класс становится слишком большим
- B. Потому что интерфейсы становятся слишком мелкими
- C. Потому что подкласс нельзя использовать вместо базового без изменения поведения ✓
- D. Потому что нарушает принцип открытости/закрытости

✓ 1/1

Какой принцип SOLID рекомендует разбивать большие интерфейсы на более мелкие?

- A. DIP
- B. SRP
- C. ISP ✓
- D. OCP

✓ 1/1

Какой принцип SOLID гласит, что высокоуровневые модули не должны зависеть от модулей низкого уровня напрямую, а оба должны зависеть от абстракций?

- A. DIP ✓
- B. LSP
- C. OCP
- D. SRP

✗ ···/1

- Вариант 1

✓ Какое утверждение иллюстрирует принцип KISS? 1/1

- A. Добавлять все возможные проверки заранее
- B. Писать сложные оптимизации для кода, который ещё не работает
- C. Выбирать простейшее рабочее решение вместо избыточной сложности ✓
- D. Оставлять дублированный код, чтобы не потерять логику

✓ Какой принцип нарушает следующий код? 1/1

```
public void dailyReport() {
    Connection c = connectDB();
    // выполнение запроса
    c.close();
}
public void weeklyReport() {
    Connection c = connectDB();
    // выполнение запроса
    c.close();
}
public void monthlyReport() {
    Connection c = connectDB();
    // выполнение запроса
    c.close();
}
```

- A. SRP
- B. DRY ✓
- C. OCP
- D. KISS

Как можно упростить следующий метод по принципу KISS?

```
public boolean isEven(int n) {
    if (n % 2 == 0) return true;
    else return false;
}
```

- [x] A. return n % 2 == 0;
- [ ] B. return (n / 2) * 2 == n;
- [ ] C. if (n % 2 != 0) return false; return true;
- [ ] D. return n % 2;

0/1

Что выведет следующий код и почему?

```
public class Test {
    public static void main(String[] args) {
        int a = 1;
        a = a + ++a;
        System.out.println(a);
    }
}
```

- A. 2
- B. 3
- C. 4
- D. 5

Richtige Antwort

- B. 3

Feedback

Объяснение: В выражении a + ++a сначала используется старое значение a (1), затем a инкрементируется до 2, и суммирование даёт 3.

0/1

Какой принцип нарушён в следующем коде?

```
interface Machine {
    void print(Document d);
    void scan(Document d);
    void fax(Document d);
}
class SimplePrinter implements Machine {
    public void print(Document d) { /* … */ }
    public void scan(Document d) { throw new UnsupportedOperationException(); }
    public void fax(Document d) { throw new UnsupportedOperationException(); }
}
```

- (●) A. SRP
- ( ) B. OCP
- ( ) C. ISP
- ( ) D. DIP

Richtige Antwort

- (●) C. ISP

Feedback

Объяснение: Класс зависит от методов, которые он не использует, нарушая принцип разделения интерфейсов.

1/1

Какое значение будет выведено и почему?

```
class Counter {
    public static void increment(int n) { n++; }
    public static void main(String[] args) {
        int a = 5;
        increment(a);
        System.out.println(a);
    }
}
```

- (●) A. 5 ✓
- ( ) B. 6
- ( ) C. Ошибка компиляции
- ( ) D. Null