# Programovanie (4)

## Cvičenie 02 - Code review

Alexander Šimko

- Code review je subjektívna záležitosť
- Ak by ju robil niekto iný, mal by na kód iný názor

# Aký kód písať?

1. Píšte kód, ktorý funguje správne (bez chýb, bez memory leakov)
2. Píšte čitateľný kód
3. Píšte efektívny kód (časová, pamäťová zložitosť)

# Priemer po 1.

## Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null) return 0;
    if (data.length == 0) return 0;
    double sucet = 0;
    for (String datum : data) {
        if (datum == null) continue;
        if (datum.isEmpty()) continue;
        if (datum.matches("^[-+]?[0-9]+$")) {
            sucet += Double.parseDouble(datum);
        }
    }
    return sucet/data.length;
}
```

## Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null) return 0;
    if (data.length == 0) return 0;
    double sucet = 0;
    for (String datum : data) {
        if (datum == null) continue;
        if (datum.isEmpty()) continue;
        if (datum.matches("^[-+]?[0-9]+$")) {
            sucet += Double.parseDouble(datum);
        }
    }
    return sucet/data.length;
}
```

Miešame angličtinu a slovenčinu

# Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null) return 0;
    if (data.length == 0) return 0;
    double sucet = 0;
    for (String datum : data) {
        if (datum == null) continue;
        if (datum.isEmpty()) continue;
        if (datum.matches("^[-+]?[0-9]+$")) {
            sucet += Double.parseDouble(datum);
        }
    }
    return sucet/data.length;
}
```

Miešame angličtinu a slovenčinu

Používajme angličtinu

```
double sum = 0;
```

## Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null) return 0;
    if (data.length == 0) return 0;
    double sucet = 0;
    for (String datum : data) {
        if (datum == null) continue;
        if (datum.isEmpty()) continue;
        if (datum.matches("^[-+]?[0-9]+$")) {
            sucet += Double.parseDouble(datum);
        }
    }
    return sucet/data.length;
}
```

# Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null) return 0;
    if (data.length == 0) return 0;
    double sucet = 0;
    for (String datum : data) {
        if (datum == null) continue;
        if (datum.isEmpty()) continue;
        if (datum.matches("^[-+]?[0-9]+$")) {
            sucet += Double.parseDouble(datum);
        }
    }
    return sucet/data.length;
}
```

Kontrolujeme dookola ten istý vzor

# Čo vieme zlepšiť?

```java
public static double priemer(String[] data) {
    if (data == null) return 0;
    if (data.length == 0) return 0;
    double sucet = 0;
    for (String datum : data) {
        if (datum == null) continue;
        if (datum.isEmpty()) continue;
        if (datum.matches("^[-+]?[0-9]+$")) {

    }
    r
}
```

Kontrolujeme dookola ten istý vzor

```java
class String {
    public boolean matches(String regex) {
        return Pattern.matches(regex, this);
    }
}
```

# Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null) return 0;
    if (data.length == 0) return 0;
    double sucet = 0;
    for (String datum : data) {
        if (datum == null) continue;
        if (datum.isEmpty()) continue;
        if (datum.matches("^[-+]?[0-9]+$")) {

    }
    r
}
```

Kontrolujeme dookola ten istý vzor

```
class String {
   public boolean matches(String regex) {
       return Pattern.matches(regex, this);
   }
}
```

```
class Pattern {
public static boolean matches(String regex, CharSequence input)
{
    Pattern p = Pattern.compile(regex);
    Matcher m = p.matcher(input);
    return m.matches();
}
}
```

Dookola sa kompiluje ten istý vzor

# Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null) return 0;
    if (data.length == 0) return 0;
    double sucet = 0;
    for (String datum : data) {
        if (datum == null) continue;
        if (datum.isEmpty()) continue;
        if (datum.matches("^[-+]?[0-9]+$")) {
            sucet += Double.parseDouble(datum);
        }
    }
```

Kontrolujeme dookola ten istý vzor

Vzor skompilujme iba raz

```
static Pattern PATTERN = Pattern.compile("^[-+]?[0-9]+$");

public static double priemer(String[] data) {
   for (String datum : data) {
      …
      if (PATTERN.matcher(datum).matches()) {
           sucet += Double.parseDouble(datum);
      }
      …
   }
 …
}
```

# Čo vieme zlepšiť?

```java
public static double priemer(String[] data) {
    if (data == null) return 0;
    if (data.length == 0) return 0;
    double sucet = 0;
    for (String datum : data) {
        if (datum == null) continue;
        if (datum.isEmpty()) continue;
        if (datum.matches("^[-+]?[0-9]+$")) {
            sucet += Double.parseDouble(datum);
        }
    }
    return sucet/data.length;
}
```

# Čo vieme zlepšiť?

```java
public static double priemer(String[] data) {
    if (data == null) return 0;
    if (data.length == 0) return 0;
    double sucet = 0;
    for (String datum : data) {
        if (datum == null) continue;
        if (datum.isEmpty()) continue;
        if (datum.matches("^[-+]?[0-9]+$")) {
            sucet += Double.parseDouble(datum);
        }
    }
    return sucet/data.length;
}
```

Najlepší je kód, ktorý nikdy nenapíšeme

```java
for (String datum : data) {
    try {
        sucet += Integer.parseInt(datum);
    } catch (NumberFormatException ignored) {
    }
}
```

## Priemer po 2.

## Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null){
        return 0;
    }
    long sucet = 0;
    double n = 0;
    for(int i = 0; i < data.length; i++) {
        try {
            sucet += Integer.valueOf(data[i]);
            n++;
        } catch (Exception e){

        }
    }
    if (n == 0) {
        return 0;
    }
    return sucet/n;
}
```

## Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null){
        return 0;
    }
    long sucet = 0;
    double n = 0;
    for(int i = 0; i < data.length; i++) {
        try {
            sucet += Integer.valueOf(data[i]);
            n++;
        } catch (Exception e){

        }
    }
    if (n == 0) {
        return 0;
    }
    return sucet/n;
}
```

null kontrolujeme cez NullPointerException

# Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null){
        return 0;
    }
    long sucet = 0;
    double n = 0;
    for(int i = 0; i < data.length; i++) {
        try {
            sucet += Integer.valueOf(data[i]);
            n++;
        } catch (Exception e){

        }
    }
    if (n == 0) {
        return 0;
    }
    return sucet/n;
}
```

null kontrolujeme cez NullPointerException

NullPointerException znamená chybu programátora

## Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null){
        return 0;
    }
    long sucet = 0;
    double n = 0;
    for(int i = 0; i < data.length; i++) {
        try {
            sucet += Integer.valueOf(data[i]);
            n++;
        } catch (Exception e){

        }
    }
    if (n
        re
    }
    return
}
```

```
String datum = data[i];
if (datum == null) {
  continue;
}

try {
    sucet += Integer.parseInt(datum);
} catch (NumberFormatException ignored) {
}
```

null vyriešme cez if

# Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null){
        return 0;
    }
    long sucet = 0;
    double n = 0;
    for(int i = 0; i < data.length; i++) {
        try {
            sucet += Integer.valueOf(data[i]);
            n++;
        } catch (Exception e){

        }
    }
    if (n == 0) {
        return 0;
    }
    return sucet/n;
}
```

Chytáme všetky výnimky

## Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null){
        return 0;
    }
    long sucet = 0;
    double n = 0;
    for(int i = 0; i < data.length; i++) {
        try {
            sucet += Integer.valueOf(data[i]);
            n++;
        } catch (Exception e){

        }
    }
    if (n
        re
    }
    return
}
```

Chytáme všetky výnimky

Chytajme iba tie, ktoré očakávame (NullPointerException riešime cez if)

```
try {
    sucet += Integer.parseInt(datum);
} catch (NumberFormatException ignored) {
}
```

## Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null){
        return 0;
    }
    long sucet = 0;
    double n = 0;
    for(int i = 0; i < data.length; i++) {
        try {
            sucet += Integer.valueOf(data[i]);
            n++;
        } catch (Exception e){

        }
    }
    if (n == 0) {
        return 0;
    }
    return sucet/n;
}
```

# Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null){
        return 0;
    }
    long sucet = 0;
    double n = 0;
    for(int i = 0; i < data.length; i++) {
        try {
            sucet += Integer.valueOf(data[i]);
            n++;
        } catch (Exception e){

        }
    }
    if (n == 0) {
        return 0;
    }
    return sucet/n;
}
```

valueOf vracia inštanciu wrapper typu Integer

## Čo vieme zlepšiť?

```
public static double priemer(String[] data) {
    if (data == null){
        return 0;
    }
    long sucet = 0;
    double n = 0;
    for(int i = 0; i < data.length; i++) {
        try {
            sucet += Integer.valueOf(data[i]);
            n++;
        } catch (Exception e){

        }
    }
    if (n == 0) {
        return 0;
    }
    return sucet/n;
}
```

valueOf vracia inštanciu wrapper typu Integer

stačí nám primitívny typ int

```
try {
    sucet += Integer.parseInt(datum);
} catch (NumberFormatException ignored) {
}
```

## Priemer po 3.

## Čo vieme zlepšiť?

```java
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        c = input.charAt(i);
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

## Čo vieme zlepšiť?

```
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        c = input.charAt(i);
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

premenná je definovaná priskoro

## Čo vieme zlepšiť?

```
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        c = input.charAt(i);
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

premenná je definovaná priskoro

definujme ju až ju potrebujeme

```
char c = input.charAt(i);
```

## Čo vieme zlepšiť?

```java
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        c = input.charAt(i);
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

## Čo vieme zlepšiť?

```java
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        c = input.charAt(i);
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

input aj “” je inštancia triedy

== porovnáva adresy objektov, nie ich obsah

# Čo vieme zlepšiť?

```
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i =
        c = inpu
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

input aj “” je inštancia triedy
== porovnáva adresy objektov, nie ich obsah

```
if ("".equals(input)) {
    return false;
}
```

## Čo vieme zlepšiť?

```java
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        c = input.charAt(i);
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

# Čo vieme zlepšiť?

```
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        c = input.charAt(i);
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

Java konvencia je mať zátvorku { na konci riadku

# Čo vieme zlepšiť?

```
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        c = input.charAt(i);
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

Java konvencia je mať zátvorku { na konci riadku

```
if ("".equals(input)) {
    return false;
}
for (int i = 0; i < input.length(); i++) {
    c = input.charAt(i);
    if (!(c >= '0' && c <= '9')) {
        return false;
    }
}
return true;
```

# Čo vieme zlepšiť?

```
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        c = input.charAt(i);
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

V každom cykle zisťujeme dĺžku stringu, hoci string je immutable a nemôže sa zmeniť

## Čo vieme zlepšiť?

```
private static boolean isNumber(String input){
    char c;
    if (input == "")
    {
        return false;
    }
    for (int i = 0; i < input.length(); i++) {
        c = input.charAt(i);
        if (!(c >= '0' && c <= '9'))
        {
            return false;
        }
    }
    return true;
}
```

V každom cykle zisťujeme dĺžku stringu, hoci string je immutable a nemôže sa zmeniť

```
int length = input.length();

for (int i = 0; i < length; i++) {
    c = input.charAt(i);
    …
}
```

Podobne po 1.

# Čo vieme zlepšiť?

```java
public static boolean podobne(String[] a, Integer[] b) {
    if (a == null && b == null) return true;
    if(a == null || b == null) return false;
    if(a.length != b.length) return false;

    int length = a.length;
    for(int i=0; i < length; i++){
        if(a[i] == null && b[i] ==null) return true;
        if(a[i] == null || b[i] ==null) return false;
        if(countA(a[i]) != b[i]) return false;
    }
    return true;
}

public static boolean podobne2(String[][] a, Integer[][] b) {
    if (a == null && b == null) return true;
    if(a == null || b == null) return false;
    if(a.length != b.length) return false;

    int length = a.length;
    for(int i=0; i < length; i++){
        if (a[i] == null && b[i] == null) continue;
        if(a[i] == null || b[i] == null) return false;
        if(a[i].length != b[i].length) return false;

        int inside_length = a[i].length;

        for(int j =0; j < inside_length; j++) {
            if (a[i][j] == null && b[i][j]  == null) continue;
            if (a[i][j]  == null || b[i][j]  == null) return false;
            if (countA(a[i][j]) != b[i][j]) return false;

        }
    }
    return true;
}
```

# Čo vieme zlepšiť?

```
public static boolean podobne(String[] a, Integer[] b) {
    if (a == null && b == null) return true;
    if(a == null || b == null) return false;
    if(a.length != b.length) return false;

    int length = a.length;
    for(int i=0; i < length; i++){
        if(a[i] == null && b[i] ==null) return true;
        if(a[i] == null || b[i] ==null) return false;
        if(countA(a[i]) != b[i]) return false;
    }
    return true;
}

public static boolean podobne2(String[][] a, Integer[][] b) {
    if (a == null && b == null) return true;
    if(a == null || b == null) return false;
    if(a.length != b.length) return false;

    int length = a.length;
    for(int i=0; i < length; i++){
        if (a[i] == null && b[i] == null) continue;
        if(a[i] == null || b[i] == null) return false;
        if(a[i].length != b[i].length) return false;

        int inside_length = a[i].length;

        for(int j =0; j < inside_length; j++) {
            if (a[i][j] == null && b[i][j]  == null) continue;
            if (a[i][j]  == null || b[i][j]  == null) return false;
            if (countA(a[i][j]) != b[i][j]) return false;

        }
    }
    return true;
}
```

Duplicitný kód

# Čo vieme zlepšiť?

```java
public static boolean podobne(String[] a, Integer[] b) {
    if (a == null && b == null) return true;
    if(a == null || b == null) return false;
    if(a.length != b.length) return false;

    int length = a.length;
    for(int i=0; i < length; i++){
        if(a[i] == null && b[i] ==null) return true;
        if(a[i] == null || b[i] ==null) return false;
        if(countA(a[i]) != b[i]) return false;
    }
    return true;
}

public static boolean podobne2(String[][] a, Integer[][] b) {
    if (a == null && b == null) return true;
    if(a == null || b == null) return false;
    if(a.length != b.length) return false;

    int length = a.length;
    for(int i=0; i < length; i++){
        if (a[i] == null && b[i]
        if(a[i] == null || b[i]
        if(a[i].length != b[i].l

        int inside_length = a[i]

        for(int j =0; j < inside
            if (a[i][j] == null
            if (a[i][j]  == null || b[i][j] == null) return false;
            if (countA(a[i][j]) != b[i][j]) return false;

        }
    }
    return true;
}
```

```java
for(int i=0; i < length; i++){
    if(podobne(a[i], b[i]) == false) {
        return false;
    }
}
```

Duplicitný kód

# Čo vieme zlepšiť?

```
public static boolean podobne(String[] a, Integer[] b) {
    if (a == null && b == null) return true;
    if(a == null || b == null) return false;
    if(a.length != b.length) return false;

    int length = a.length;
    for(int i=0; i < length; i++){
        if(a[i] == null && b[i] ==null) return true;
        if(a[i] == null || b[i] ==null) return false;
        if(countA(a[i]) != b[i]) return false;
    }
    return true;
}

public static boolean podobne2(String[][] a, Integer[][] b) {
    if (a == null && b == null) return true;
    if(a == null || b == null) return false;
    if(a.length != b.length) return false;

    int length = a.length;
    for(int i=0; i < length; i++){
        if (a[i] == null && b[i] == null) continue;
        if(a[i] == null || b[i] == null) return false;
        if(a[i].length != b[i].length) return false;

        int inside_length = a[i].length;

        for(int j =0; j < inside_length; j++) {
            if (a[i][j] == null && b[i][j]  == null) continue;
            if (a[i][j]  == null || b[i][j]  == null) return false;
            if (countA(a[i][j]) != b[i][j]) return false;

        }
    }
    return true;
}
```

# Čo vieme zlepšiť?

Duplicitný kód

```
public static boolean podobne(String[] a, Integer[] b) {
    if (a == null && b == null) return true;
    if(a == null || b == null) return false;
    if(a.length != b.length) return false;

    int length = a.length;
    for(int i=0; i < length; i++){
        if(a[i] == null && b[i] ==null) return true;
        if(a[i] == null || b[i] ==null) return false;
        if(countA(a[i]) != b[i]) return false;
    }
    return true;
}

public static boolean podobne2(String[][] a, Integer[][] b) {
    if (a == null && b == null) return true;
    if(a == null || b == null) return false;
    if(a.length != b.length) return false;

    int length = a.length;
    for(int i=0; i < length; i++){
        if (a[i] == null && b[i] == null) continue;
        if(a[i] == null || b[i] == null) return false;
        if(a[i].length != b[i].length) return false;

        int inside_length = a[i].length;

        for(int j =0; j < inside_length; j++) {
            if (a[i][j] == null && b[i][j]  == null) continue;
            if (a[i][j]  == null || b[i][j]  == null) return false;
            if (countA(a[i][j]) != b[i][j]) return false;

        }
    }
    return true;
}
```

# Čo vieme zlepšiť?

```java
private static Boolean haveSameStructure(Object[] a, Object[] b) {
    if (a == null && b == null) return Boolean.TRUE;
    if(a == null || b == null) return Boolean.FALSE;
    if(a.length != b.length) return Boolean.FALSE;
    return null;
}

public static boolean podobne(String[] a, Integer[] b) {
    Boolean sameStructure = haveSameStructure(a, b);
    if (sameStructure != null) {
        return sameStructure;
    }

    int length = a.length;
    for(int i=0; i < length; i++){
   …
}

public static boolean podobne2(String[][] a, Integer[][] b) {
    Boolean sameStructure = haveSameStructure(a, b);
    if (sameStructure != null) {
        return sameStructure;
    }

    int length = a.length;
    for(int i=0; i < length; i++){
   …
}
```

## Alternating po 1.

## Čo vieme zlepšiť?

```java
public static int count(String str, char first, char second) {
    int count = 0;
    char lastChar = 's';
    boolean valid = false;
    int length = str.length();
    for (int i = 0; i < length; i++) {
        char ch = str.charAt(i);
        if (ch == first || ch == second) {
            if (lastChar != ch || valid == false) {
                valid = true;
                lastChar = ch;
                count++;
            } else {
                return 0;
            }
        }
    }
    return count;
}
```

# Čo vieme zlepšiť?

```
public static int count(String str, char first, char second) {
    int count = 0;
    char lastChar = 's';
    boolean valid = false;
    int length = str.length();
    for (int i = 0; i < length; i++) {
        char ch = str.charAt(i);
        if (ch == first || ch == second) {
            if (lastChar != ch || valid == false) {
                valid = true;
                lastChar = ch;
                count++;
            } else {
                return 0;
            }
        }
    }
    return count;
}
```

Názvy

# Čo vieme zlepšiť?

```java
public static int count(String str, char first, char second) {
    int count = 0;
    char lastChar = 's';
```

Názvy

```java
static int computeAlternatingSequenceLength(String sequence, char
firstChar, char secondChar) {
    int alternatingLength = 0;
    char lastChar = 's';
    boolean isLastCharValid = false;
    int length = sequence.length();
    for (int i = 0; i < length; i++) {
        char currentChar = sequence.charAt(i);
        if (currentChar == firstChar || currentChar == secondChar) {
            if (lastChar != currentChar || isLastCharValid == false) {
                isLastCharValid = true;
                lastChar = currentChar;
                alternatingLength++;
            } else {
                return 0;
            }
        }
    }
    return alternatingLength;
}
```

## Čo vieme zlepšiť?

```java
static int computeAlternatingSequenceLength2(String sequence, char
firstChar, char secondChar) {
    int alternatingLength = 0;
    char lastChar = 's';
    boolean isLastCharValid = false;
    int length = sequence.length();
    for (int i = 0; i < length; i++) {
        char currentChar = sequence.charAt(i);
        if (currentChar == firstChar || currentChar == secondChar) {
            if (lastChar != currentChar || isLastCharValid == false) {
                isLastCharValid = true;
                lastChar = currentChar;
                alternatingLength++;
            } else {
                return 0;
            }
        }
    }
    return alternatingLength;
}
```

## Čo vieme zlepšiť?

```
static int computeAlternatingSequenceLength2(String sequence, char
firstChar, char secondChar) {
    int alternatingLength = 0;
    char lastChar = 's';
    boolean isLastCharValid = false;
    int length = sequence.length();
    for (int i = 0; i < length; i++) {
        char currentChar = sequence.charAt(i);
        if (currentChar == firstChar || currentChar == secondChar) {
            if (lastChar != currentChar || isLastCharValid == false) {
                isLastCharValid = true;
                lastChar = currentChar;
                alternatingLength++;
            } else {
                return 0;
            }
        }
    }
    return alternatingLength;
}
```

3 úrovne vnorenia

## Čo vieme zlepšiť?

```
static int computeAlternatingSequenceLength2(String sequence, char
firstChar, char secondChar) {
    int alternatingLength = 0;
    char lastChar = 's';
    boolean isLastCharValid = false;
    int length = sequence.length();
    for (int i = 0; i < length; i++) {
        char currentChar = sequence.charAt(i);
        if (currentChar == firstChar || currentChar == secondChar) {
    }
    re
}
```

3 úrovne vnorenia

```
for (int i = 0; i < length; i++) {
    char currentChar = sequence.charAt(i);
    if (currentChar != firstChar && currentChar != secondChar) {
        continue;
    }

    if (lastChar != currentChar || isLastCharValid == false) {
        isLastCharValid = true;
        lastChar = currentChar;
        alternatingLength++;
    } else {
        return 0;
    }
}
```

# Čo vieme zlepšiť?

```
static int computeAlternatingSequenceLength2(String sequence, char
firstChar, char secondChar) {
    int alternatingLength = 0;
    char lastChar = 's';
    boolean isLastCharValid = false;
    int length = sequence.length();
    for (int i = 0; i < length; i++) {
        char currentChar = sequence.charAt(i);
        if (currentChar == firstChar || currentChar == secondChar) {

    }
    re
}
```

3 úrovne vnorenia

```
for (int i = 0; i < length; i++) {
    char currentChar = sequence.charAt(i);
    if (currentChar != firstChar && currentChar != secondChar) {
        continue;
    }

    if (lastChar != currentChar || isLastCharValid == false) {
        isLastCharValid = true;
        lastChar = currentChar;
        alternatingLength++;
    } else {
        return 0;
    }
}
```

Tento krok je kontroverzný, niekomu sa nemusí páčiť continue

## Alternating po 2.

# Čo vieme zlepšiť?

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = novy.length();
    for (int i = 0; i < dlzka; i++) {
        char first = novy.charAt(i);
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

# Čo vieme zlepšiť?

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = novy.length();
    for (int i = 0; i < dlzka; i++) {
        char first = novy.charAt(i);
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

Názov má komunikovať účel

# Čo vieme zlepšiť?

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = novy.length();
    for (int i = 0; i < dlzka; i++) {
        char first = novy.charAt(i);
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

**Názov má komunikovať účel**

```
StringBuffer chars = new StringBuffer();
```

## Čo vieme zlepšiť?

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = novy.length();
    for (int i = 0; i < dlzka; i++) {
        char first = novy.charAt(i);
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

# Čo vieme zlepšiť?

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = novy.length();
    for (int i = 0; i < dlzka; i++) {
        char first = novy.charAt(i);
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

StringBuffer ak k nemu pristupujeme z viacerých vlákien

# Čo vieme zlepšiť?

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = novy.length();
    for (int i = 0; i < dlzka; i++) {
        char first = novy.charAt(i);
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

**StringBuffer ak k nemu pristupujeme z viacerých vlákien**

**Ináč StringBuilder**

```
StringBuilder chars = new StringBuilder();
```

## Čo vieme zlepšiť?

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = novy.length();
    for (int i = 0; i < dlzka; i++) {
        char first = novy.charAt(i);
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

## Čo vieme zlepšiť?

Magické konštanty

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = novy.length();
    for (int i = 0; i < dlzka; i++) {
        char first = novy.charAt(i);
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

# Čo vieme zlepšiť?

```java
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = no
    for (int i = 0
        char first
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

Magické konštanty

```java
for (char i = 'a'; i <= 'z'; i++) {
    if (str.contains("" + i)) {
        novy.append(i);
    }
}
```

# Čo vieme zlepšiť?

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = no
    for (int i = 0
        char first
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

Magické konštanty

```
for (char i = 'a'; i <= 'z'; i++) {
    if (str.contains("" + i)) {
        novy.append(i);
    }
}
```

Zbytočne vytvárame inštanciu stringu

# Čo vieme zlepšiť?

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = no
    for (int i = 0
        char first
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

Magické konštanty

```
for (char i = 'a'; i <= 'z'; i++) {
    if (str.contains("" + i)) {
        novy.append(i);
    }
}
```

Zbytočne vytvárame inštanciu stringu

```
for (char i = 'a'; i <= 'z'; i++) {
    if (str.indexOf(i) != -1) {
        novy.append(i);
    }
}
```

# Čo vieme zlepšiť?

```java
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = no
    for (int i = 0
        char first
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

Magické konštanty

```java
for (char i = 'a'; i <= 'z'; i++) {
    if (str.contains("" + i)) {
        novy.append(i);
    }
}
```

Zbytočne vytvárame inštanciu stringu

```java
for (char i = 'a'; i <= 'z'; i++) {
    if (str.indexOf(i) != -1) {
        novy.append(i);
    }
}
```

V testoch bola iba latinka, ale zadanie to negarantovalo

# Čo vieme zlepšiť?

```
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = novy.length();
    for (int i = 0; i < dlzka; i++) {
        char first = novy.charAt(i);
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

# Čo vieme zlepšiť?

```java
public static int alternate(String str){
    StringBuffer novy = new StringBuffer("");

    for (int i=0;i<26;i++){
        if(str.contains(""+(char) (i + (int) 'a'))){
            novy.append((char) (i + (int) 'a'));
        }
    }
    int maxi = 0;
    int dlzka = novy.length();
    for (int i = 0; i < dlzka; i++) {
        char first = novy.charAt(i);
        for (int j = i+1; j < dlzka; j++) {
            int count = cunt(str, first, novy.charAt(j));
            maxi = Math.max(count, maxi);
        }
    }
    return maxi;
}
```

Hmmm ?

## Alternating po 3.

# Čo vieme zlepšiť?

```java
public static int alternate(String str) {
    //System.out.println(str);
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
        }
    }
    for (int i = 0; i < letters.length() - 1; i++) {
        for (int k = i + 1; k < letters.length(); k++) {
            String regexp = "[^" + letters.toString().charAt(i) + letters.toString().charAt(k) + "]";
            Pattern pattern = Pattern.compile(regexp);
            String strCopy = str.replaceAll(pattern.pattern(), "");
            char first = strCopy.charAt(0);
            char sec = strCopy.charAt(1);
            if (first == sec) continue;
            //regexp = String.format("(\\w\\w)\\1{%d}", (int) ((strCopy.length() / 2) - 1));
            //pattern = Pattern.compile(regexp);
            boolean one = false;
            //System.out.println(strCopy);
            if (strCopy.length() % 2 == 1) {
                if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                    continue;
                }
                else {
                    strCopy = strCopy.substring(0, strCopy.length() - 1);
                    one = true;
                }
            }
            boolean valid = true;
            for (int j = 2; j < strCopy.length() - 1; j += 2) {
                if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                    valid = false;
                    break;
                }
            }
            if (valid && strCopy.length() > max) {
                System.out.println(str);
                System.out.println(strCopy);
                max = (strCopy.length() + (one ? 1 : 0));
            }
            //if ((strCopy.length() == 2 || strCopy.matches(pattern.pattern())) && max < (strCopy.length() + (one ? 1 : 0))) {
            //    max = (strCopy.length() + (one ? 1 : 0));
            //}
        }
    }
    return max;
```

# Čo vieme zlepšiť?

Komentáre slúžia na objasnenie živého kódu,
nie na vyradenie mŕtveho kódu

```java
public static int alternate(String str) {
    //System.out.println(str);
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
        }
    }
    for (int i = 0; i < letters.length() - 1; i++) {
        for (int k = i + 1; k < letters.length(); k++) {
            String regexp = "[^" + letters.toString().charAt(i) + letters.toString().charAt(k) + "]";
            Pattern pattern = Pattern.compile(regexp);
            String strCopy = str.replaceAll(pattern.pattern(), "");
            char first = strCopy.charAt(0);
            char sec = strCopy.charAt(1);
            if (first == sec) continue;
            //regexp = String.format("(\\w\\w)\\1{%d}", (int) ((strCopy.length() / 2) - 1));
            //pattern = Pattern.compile(regexp);
            boolean one = false;
            //System.out.println(strCopy);
            if (strCopy.length() % 2 == 1) {
                if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                    continue;
                }
                else {
                    strCopy = strCopy.substring(0, strCopy.length() - 1);
                    one = true;
                }
            }
            boolean valid = true;
            for (int j = 2; j < strCopy.length() - 1; j += 2) {
                if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                    valid = false;
                    break;
                }
            }
            if (valid && strCopy.length() > max) {
                System.out.println(str);
                System.out.println(strCopy);
                max = (strCopy.length() + (one ? 1 : 0));
            }
            //if ((strCopy.length() == 2 || strCopy.matches(pattern.pattern())) && max < (strCopy.length() + (one ? 1 : 0))) {
            //    max = (strCopy.length() + (one ? 1 : 0));
            //}
        }
    }
    return max;
```

# Čo vieme zlepšiť?

Komentáre slúžia na objasnenie živého kódu, nie na vyradenie mŕtveho kódu

```java
public static int alternate2(String str) {
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
        }
    }
    for (int i = 0; i < letters.length() - 1; i++) {
        for (int k = i + 1; k < letters.length(); k++) {
            String regexp = "[^" + letters.toString().charAt(i) + letters.toString().charAt(k) + "]";
            Pattern pattern = Pattern.compile(regexp);
            String strCopy = str.replaceAll(pattern.pattern(), "");
            char first = strCopy.charAt(0);
            char sec = strCopy.charAt(1);
            if (first == sec) continue;

            boolean one = false;

            if (strCopy.length() % 2 == 1) {
                if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                    continue;
                }
                else {
                    strCopy = strCopy.substring(0, strCopy.length() - 1);
                    one = true;
                }
            }
            boolean valid = true;
            for (int j = 2; j < strCopy.length() - 1; j += 2) {
                if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                    valid = false;
                    break;
                }
            }
            if (valid && strCopy.length() > max) {
                System.out.println(str);
                System.out.println(strCopy);
                max = (strCopy.length() + (one ? 1 : 0));
            }
        }
    }
```

# Čo vieme zlepšiť?

Metóda je dlhá

```
public static int alternate2(String str) {
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
        }
    }
    for (int i = 0; i < letters.length() - 1; i++) {
        for (int k = i + 1; k < letters.length(); k++) {
            String regexp = "[^" + letters.toString().charAt(i) + letters.toString().charAt(k) + "]";
            Pattern pattern = Pattern.compile(regexp);
            String strCopy = str.replaceAll(pattern.pattern(), "");
            char first = strCopy.charAt(0);
            char sec = strCopy.charAt(1);
            if (first == sec) continue;

            boolean one = false;

            if (strCopy.length() % 2 == 1) {
                if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                    continue;
                }
                else {
                    strCopy = strCopy.substring(0, strCopy.length() - 1);
                    one = true;
                }
            }
            boolean valid = true;
            for (int j = 2; j < strCopy.length() - 1; j += 2) {
                if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                    valid = false;
                    break;
                }
            }
            if (valid && strCopy.length() > max) {
                System.out.println(str);
                System.out.println(strCopy);
                max = (strCopy.length() + (one ? 1 : 0));
            }
        }
    }
```

# Čo vieme zlepšiť?

```
public static int alternate2(String str) {
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
```

Metóda je dlhá

```
static StringBuilder extractLetters(String str) {
…
}

static boolean isAlternating(String str) {
…
}

public static int alternate(String str) {
    StringBuilder letters = extractLetters(str);
    for (int i = 0; i < letters.length() - 1; i++) {
        for (int k = i + 1; k < letters.length(); k++) {
            String regexp = …
            String strCopy = str.replaceAll(regexp, "");
            if (isAlternating(strCopy) && strCopy.length() > max) {
                max = strCopy.length();
            }
        }
    }
    return max;
}
```

```
        }
    }
```

# Čo vieme zlepšiť?

```
public static int alternate(String str) {
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
        }
    }
```

# Čo vieme zlepšiť?

```
public static int alternate(String str) {
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
        }
    }
```

V každej iterácii zisťujeme dĺžku, ktorá sa nemení

# Čo vieme zlepšiť?

```
public static int alternate(String str) {
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
        }
    }
```

**V každej iterácii zisťujeme dĺžku, ktorá sa nemení**

```
int length = str.length();
for (int i = 0; i < length; i++) {
   …
}
```

## Čo vieme zlepšiť?

```java
public static int alternate(String str) {
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
        }
    }
```

## Čo vieme zlepšiť?

```
public static int alternate(String str) {
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
        }
    }
```

Vytváranie nových stringov v takomto jednoduchom prípade je nevhodné

# Čo vieme zlepšiť?

```
public static int alternate(String str) {
    int max = 0;
    StringBuilder letters = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (!letters.toString().contains("" + str.charAt(i))) {
            letters.append(str.charAt(i));
        }
    }
```

Vytváranie nových stringov v takomto jednoduchom prípade je nevhodné

```
int length = str.length();
for (int i = 0; i < length; i++) {
    if (!contains(str, str.charAt(i)) {
        letters.append(str.charAt(i));
    }
}
```

# Čo vieme zlepšiť?

```java
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
        String regexp = "[^" + letters.toString().charAt(i) + letters.toString().charAt(k) + "]";
        Pattern pattern = Pattern.compile(regexp);
        String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;

        boolean one = false;

        if (strCopy.length() % 2 == 1) {
            if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                continue;
            }
            else {
                strCopy = strCopy.substring(0, strCopy.length() - 1);
                one = true;
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
            if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                valid = false;
                break;
            }
        }
        if (valid && strCopy.length() > max) {
            System.out.println(str);
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
        String regexp = "[^" + letters.toString().charAt(i) + letters.toString().charAt(k) + "]";
        Pattern pattern = Pattern.compile(regexp);
        String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;

        boolean one = false;

        if (strCopy.length() % 2 == 1) {
            if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                continue;
            }
            else {
                strCopy = strCopy.substring(0, strCopy.length() - 1);
                one = true;
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
            if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                valid = false;
                break;
            }
        }
        if (valid && strCopy.length() > max) {
            System.out.println(str);
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

regexp.equals(pattern.pattern()) == true

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
        String regexp = "[^" + letters.toString().charAt(i) + letters.toString().charAt(k) + "]";
        Pattern pattern = Pattern.compile(regexp);
        String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;

        boolean one = false;
```

regexp.equals(pattern.pattern()) == true

```
String regexp = "[^" + letters.toString().charAt(i) +
letters.toString().charAt(k) + "]";
String strCopy = str.replaceAll(regexp, "");
char first = strCopy.charAt(0);
char sec = strCopy.charAt(1);
```

```
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
            if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                valid = false;
                break;
            }
        }
        if (valid && strCopy.length() > max) {
            System.out.println(str);
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

# Čo vieme zlepšiť?

```java
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
        String regexp = "[^" + letters.toString().charAt(i) + letters.toString().charAt(k) + "]";
        Pattern pattern = Pattern.compile(regexp);
        String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;

        boolean one = false;
```

**regexp.equals(pattern.pattern()) == true**

```java
String regexp = "[^" + letters.toString().charAt(i) +
letters.toString().charAt(k) + "]";
String strCopy = str.replaceAll(regexp, "");
char first = strCopy.charAt(0);
char sec = strCopy.charAt(1);
```

**V každom cykle vytvoríme 2 nové kópie**

```java
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
            if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                valid = false;
                break;
            }
        }
        if (valid && strCopy.length() > max) {
            System.out.println(str);
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
        String regexp = "[^" + letters.toString().charAt(i) + letters.toString().charAt(k) + "]";
        Pattern pattern = Pattern.compile(regexp);
        String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;

        boolean one = false;
```

regexp.equals(pattern.pattern()) == true

```
String regexp = "[^" + letters.toString().charAt(i) +
letters.toString().charAt(k) + "]";
String strCopy = str.replaceAll(regexp, "");
char first = strCopy.charAt(0);
char sec = strCopy.charAt(1);
```

V každom cykle vytvoríme 2 nové kópie

```
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
```

```
String regexp = "[^" + letters.charAt(i) + letters.charAt(k) + "]";
String strCopy = str.replaceAll(regexp, "");
char first = strCopy.charAt(0);
char sec = strCopy.charAt(1);
```

```
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
        String regexp = "[^" + letters.toString().charAt(i) + letters.toString().charAt(k) + "]";
        Pattern pattern = Pattern.compile(regexp);
        String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;

        boolean one = false;
```

regexp.equals(pattern.pattern()) == true

```
String regexp = "[^" + letters.toString().charAt(i) +
letters.toString().charAt(k) + "]";
String strCopy = str.replaceAll(regexp, "");
char first = strCopy.charAt(0);
char sec = strCopy.charAt(1);
```

V každom cykle vytvoríme 2 nové kópie

```
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
```

```
String regexp = "[^" + letters.charAt(i) + letters.charAt(k) + "]";
String strCopy = str.replaceAll(regexp, "");
char first = strCopy.charAt(0);
char sec = strCopy.charAt(1);
```

Dá sa to aj bez replace iterovaním pôvodného reťazca

```
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
       …
       String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;

        boolean one = false;

        if (strCopy.length() % 2 == 1) {
            if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                continue;
            }
            else {
                strCopy = strCopy.substring(0, strCopy.length() - 1);
                one = true;
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
            if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                valid = false;
                break;
            }
        }
        if (valid && strCopy.length() > max) {
            System.out.println(str);
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
       …
       String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;
        
        boolean one = false;
        
        if (strCopy.length() % 2 == 1) {
            if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                continue;
            }
            else {
                strCopy = strCopy.substring(0, strCopy.length() - 1);
                one = true;
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
            if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                valid = false;
                break;
            }
        }
        if (valid && strCopy.length() > max) {
            System.out.println(str);
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

Špeciálne riešime prípad zhody prvých 2 znakov

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
        …
        String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;

        boolean one = false;

        if (strCopy.length() % 2 == 1) {
            if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                continue;
            }
            else {
                strCopy = strCopy.substring(0, strCopy.length() - 1);
                one = true;
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
            if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                valid = false;
                break;
            }
        }
        if (valid && strCopy.length() > max) {
            System.out.println(str);
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

**Špeciálne riešime prípad zhody prvých 2 znakov**

**Špeciálne riešime nepárnu dĺžku**

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
       …
       String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;

        boolean one = false;

        if (strCopy.length() % 2 == 1) {
            if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                continue;
            }
            else {
                strCopy = strCopy.substring(0, strCopy.length() - 1);
                one = true;
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
            if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                valid = false;
                break;
            }
        }
        if (valid && strCopy.length() > max) {
            System.out.println(str);
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

Špeciálne riešime prípad zhody prvých 2 znakov

Špeciálne riešime nepárnu dĺžku

A potom zvyšok

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
       …
       String strCopy = str.replaceAll(pattern.pattern(), "");
        char first = strCopy.charAt(0);
        char sec = strCopy.charAt(1);
        if (first == sec) continue;

        boolean one = false;

        if (strCopy.length() % 2 == 1) {
            if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                continue;
            }
            else {
                strCopy = strCopy.substring(0, strCopy.length() - 1);
                one = true;
            }
        }
        boolean valid = true;
        for (int j = 2; j < strCopy.length() - 1; j += 2) {
            if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                valid = false;
                break;
            }
        }
        if (valid && strCopy.length() > max) {
            System.out.println(str);
            System.out.println(strCopy);
            max = (strCopy.length() + (one ? 1 : 0));
        }
    }
}
return max;
```

Špeciálne riešime prípad zhody prvých 2 znakov

Špeciálne riešime nepárnu dĺžku

A potom zvyšok

A ešte špeciálne riešime prípad nepárnej postupnosti s rôznymi poslednými dvomi znakmi

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
        …
        String strCopy = str.replaceAll(pattern.pattern(), "");
         char first = strCopy.charAt(0);
         char sec = strCopy.charAt(1);
         if (first == sec) continue;

         boolean one = false;

         if (strCopy.length() % 2 == 1) {
             if (strCopy.charAt(strCopy.length() - 1) == strCopy.charAt(strCopy.length() - 2)) {
                 continue;
             }
             else {
                 str
                 one
             }
         }
         boolean val
         for (int j =
             if (!(strCopy.charAt(j) == first && strCopy.charAt(j + 1) == sec)) {
                 valid = false;
                 break;
             }
         }
         if (valid && strCopy.length() > max) {
             System.out.println(str);
             System.out.println(strCopy);
             max = (strCopy.length() + (one ? 1 : 0));
         }
    }
}
return max;
```

Špeciálne riešime prípad zhody prvých 2 znakov

nu dĺžku

**Príliš veľa špeciálnych prípadov**

A potom zvyšok

A ešte špeciálne riešime prípad nepárnej postupnosti s rôznymi poslednými dvomi znakmi

# Čo vieme zlepšiť?

```
for (int i = 0; i < letters.length() - 1; i++) {
    for (int k = i + 1; k < letters.length(); k++) {
```

```
static int computeAlternatingSequenceLength(String sequence, char
firstChar, char secondChar) {

    int alternatingLength = 0;
    char lastChar = 's';
    boolean isLastCharValid = false;
    int length = sequence.length();
    for (int i = 0; i < length; i++) {
        char currentChar = sequence.charAt(i);
        if (currentChar != firstChar && currentChar != secondChar) {
            continue;
        }

        if (lastChar != currentChar || isLastCharValid == false) {
            isLastCharValid = true;
            lastChar = currentChar;
            alternatingLength++;
        } else {
            return 0;
        }
    }
    return alternatingLength;
}
```

ov

```
2)) {
}
r
```

## Alternating po 4.

# Generátor r-prvkových podmnožín

```java
public static List<int[]> generate(int n, int r) {
    List<int[]> combinations = new ArrayList<>();
    int[] combination = new int[r];

    // initialize with lowest lexicographic combination
    for (int i = 0; i < r; i++) {
        combination[i] = i;
    }

    while (combination[r - 1] < n) {
        combinations.add(combination.clone());

        // generate next combination in lexicographic order
        int t = r - 1;
        while (t != 0 && combination[t] == n - r + t) {
            t--;
        }
        combination[t]++;
        for (int i = t + 1; i < r; i++) {
            combination[i] = combination[i - 1] + 1;
        }
    }

    return combinations;
}
```

# A potom iterujeme 2-prvkovými množinami

```java
public static int alternate(String str) {
    if(str == null){
        return 0;
    }
    HashSet<String> resultSet=new HashSet<String>();

    for (int i = 0; i < str.length(); i++) {
        resultSet.add(Character.toString(str.charAt(i)));
    }
    int size = resultSet.size();

    List<String> chars = new ArrayList<String>(resultSet);
    List<int[]> list = generate(size, 2);
    int max = 0;
    StringBuilder new_str = new StringBuilder();

    for(int i=0; i< list.size(); i++){
        char one = chars.get(list.get(i)[0]).charAt(0);
        char two = chars.get(list.get(i)[1]).charAt(0);


      …
    }
    return max;
}
```

# A potom iterujeme 2-prvkovými množinami

```java
public static int alternate(String str) {
    if(str == null){
        return 0;
    }
    HashSet<String> resultSet=new HashSet<String>();

    for (int i = 0; i < str.length(); i++) {
        resultSet.add(Character.toString(str.charAt(i)));
    }
    int size = resultSet.size();

    List<String> chars = new ArrayList<String>(resultSet);
    List<int[]> list = generate(size, 2);
    int max = 0;
    StringBuilder new_str = new StringBuilder();

    for(int i=0; i< list.size(); i++){
        char one = chars.get(list.get(i)[0]).charAt(0);
        char two = chars.get(list.get(i)[1]).charAt(0);

     …
    }
    return max;
}
```

## A potom iterujeme 2-prvkovými množinami

```
public static int alternate(String str) {
    if(str == null){
        return 0;
    }
    HashSet<String> resultSet=new HashSet<String>();

    for (int i = 0; i < str.length(); i++) {
        resultSet.add(Character.toString(str.charAt(i)));
    }
    int size = resultSet.size();

    List<String> chars = new ArrayList<String>(resultSet);
    List<int[]> list = generate(size, 2);
    int max = 0;
    StringBuilder new_str = new StringBuilder();

    for(int i=0; i< list.size(); i++){
        char one = chars.get(list.get(i)[0]).charAt(0);
        char two = chars.get(list.get(i)[1]).charAt(0);

     …
    }
    return max;
}
```

V každom momente nám stačí jedna množina

# Držíme si iba aktuálnu množinu

```
static int[] generateBeforeFirst(int r) {
  // vráti špeciálnu r-ticu predstavujúcu “pred prvou množinou”:
  // 0, 1, 2, …, r-2, r-2
}

static bool moveToNext(int[] set, int n) {
  // set zmeň na ďalšiu množinu v poradí ak existuje
  // vráti true/false, podľa toho, či množina existuje
}

public static int alternate(String str) {
    …
    int[] set = generateBeforeFirst(2);
    int max = 0;
    StringBuilder new_str = new StringBuilder();

    while(moveToNext(set, size)){
        char one = chars.get(set[0]).charAt(0);
        char two = chars.get(set[1]).charAt(0);


     …
    }
    return max;
}
```

## Držíme si iba aktuálnu množinu

```
static int[] generateBeforeFirst(int r) {
  // vráti špeciálnu r-ticu predstavujúcu “pred prvou množinou”:
  // 0, 1, 2, …, r-2, r-2
}

static bool moveToNext(int[] set, int n) {
  // set zmeň na ďalšiu množinu v poradí ak existuje
  // vráti true/false, podľa toho, či množina existuje
}

public static int alternate(String str) {
    …
    int[] set = generateBeforeFirst(2);
    int max = 0;
    StringBuilder new_str = new StringBuilder();

    while(moveToNext(set, size)){
        char one = chars.get(set[0]).charAt(0);
        char two = chars.get(set[1]).charAt(0);


     …
    }
    return max;
}
```

Mať všeobecný generátor je pekné.
Ale vždy generujeme iba dvojprvkové množiny

# Držíme si iba aktuálnu množinu

```
static int[] generateBeforeFirst(int r) {
  // vráti špeciálnu r-ticu predstavujúcu “pred prvou množinou”:
  // 0, 1, 2, …, r-2, r-2
}

static bool moveToNext(int[] set, int n) {
  // set zmeň na ďalšiu množinu v poradí ak existuje
  // vráti true/false, podľa toho, či množina existuje
}
```

**Mať všeobecný generátor je pekné.**
**Ale vždy generujeme iba dvojprvkové množiny**

```
public static int alternate(String str) {
    …
    int[] set = generateBeforeFirst(2);
    int max = 0;
    StringBui

    while(mov
        char
        char

     …
    }
    return max;
}
```

```
int n = resultSet.size();

for (int i = 0; i < n; ++i) {
    for (int j = i + 1; j < n; ++i) {
        // mame mnozinu {i,j}
    }
}
```

## Zbieranie znakov

```
public static int alternate(String str) {
    if(str == null){
        return 0;
    }
    HashSet<String> resultSet=new HashSet<String>();

    for (int i = 0; i < str.length(); i++) {
        resultSet.add(Character.toString(str.charAt(i)));
    }

}
```

# Zbieranie znakov

```
public static int alternate(String str) {
    if(str == null){
        return 0;
    }
    HashSet<String> resultSet=new HashSet<String>();

    for (int i = 0; i < str.length(); i++) {
        resultSet.add(Character.toString(str.charAt(i)));
    }

}
```

Chceme množinu znakov, nie reťazcov

## Zbieranie znakov

```
public static int alternate(String str) {
    if(str == null){
        return 0;
    }
    HashSet<String> resultSet=new HashSet<String>();

    for (int i = 0; i < str.length(); i++) {
        resultSet.add(Character.toString(str.charAt(i)));
    }

}
```

Chceme množinu znakov, nie reťazcov

Každý znak konvertujeme na reťazec

# Zbieranie znakov

```
public static int alternate(String str) {
    if(str == null){
        return 0;
    }
    HashSet<String> resultSet=new HashSet<String>();

    for (int i = 0; i < str.length(); i++) {
        resultSet.add(Character.toString(str.charAt(i)));
    }

}
```

Chceme množinu znakov, nie reťazcov

Každý znak konvertujeme na reťazec

```
HashSet<Character> resultSet=new HashSet<>();

for (int i = 0; i < str.length(); i++) {
    resultSet.add(str.charAt(i));
}
```

Words po 1.

# Vidíme niečo tu?

```
static public boolean palindrom(String str) {
    int left = 0, right = str.length() - 1;
    char a, b;
    while (left < right) {
        while (!isLetter(str.charAt(left))) left++;
        while (!isLetter(str.charAt(right))) right--;
        if (!compareLetters(str, left, right)) {
            return false;
        }
        left++;
        right--;
    }
    return true;
}
```

## Pozor na okrajové prípady

```
static public boolean palindrom(String str) {
    int left = 0, right = str.length() - 1;
    char a, b;
    while (left < right) {
        while (!isLetter(str.charAt(left))) left++;
        while (!isLetter(str.charAt(right))) right--;
        if (!compareLetters(str, left, right)) {
            return false;
        }
        left++;
        right--;
    }
    return true;
}
```

## Pozor na okrajové prípady

```
static public boolean palindrom(String str) {
    int left = 0, right = str.length() - 1;
    char a, b;
    while (left < right) {
        while (!isLetter(str.charAt(left))) left++;
        while (!isLetter(str.charAt(right))) right--;
        if (!compareLetters(str, left, right)) {
            return false;
        }
}
```

```
public static void main(String[] args) {
    System.out.print(palindrom("12"));
}

Exception in thread "main"
java.lang.StringIndexOutOfBoundsException: String index out of
range: 2
   at java.base/
java.lang.StringLatin1.charAt(StringLatin1.java:48)
   at java.base/java.lang.String.charAt(String.java:1515)
   at Words5.palindrom(Words5.java:25)
   at Words5.main(Words5.java:19)
```

## Najjednoduchšie je najprv reťazec normalizovať

```java
static public String normalize(String str){
  // vrat retazec iba s pismenami
}
static public boolean palindrom(String str){
    String normalized = normalize(str);
    int i = 0, j = normalized.length() - 1;
    while (i < j) {
        char leftChar = Character.toLowerCase(normalized.charAt(i));
        char rightChar= Character.toLowerCase(normalized.charAt(j));
        if (leftChar != rightChar) {
            return false;
        }
        ++i;
        --j;
    }
    return true;
}
```

# Čo ak sa chceme alokácii vyhnúť?

```
static public boolean palindrom(String str) {
    int left = 0, right = str.length() - 1;
    char a, b;
    while (left < right) {
        while (!isLetter(str.charAt(left))) left++;
        while (!isLetter(str.charAt(right))) right--;
        if (!compareLetters(str, left, right)) {
            return false;
        }
        left++;
        right--;
    }
    return true;
}
```

# Čo ak sa chceme alokácii vyhnúť?

```java
static int nextLetter(String str, int index) {
    int length = str.length();
    while (++index < length) {
        if (Character.isLetter(str.charAt(index))) {
            return index;
        }
    }
    return index;
}

static int previousLetter(String str, int index) {
    while (--index >= 0) {
        if (Character.isLetter(str.charAt(index))) {
            return index;
        }
    }
    return index;
}
static public boolean palindrom(String str) {
    int i = -1, j = str.length();
    while ((i = nextLetter(str, i)) < (j = previousLetter(str, j))) {
        char leftChar = Character.toLowerCase(str.charAt(i));
        char rightChar = Character.toLowerCase(str.charAt(j));
        if (leftChar != rightChar) {
            return false;
        }
    }
    return true;
}
```

# Čo ak sa chceme alokácii vyhnúť?

```
static int nextLetter(String str, int index) {
    int length = str.length();
    while (++index < length) {
        if (Character.isLetter(str.charAt(index))) {
            return index;
        }
    }
    return index;
}

static int previousLetter(String str, int index) {
    while (--index >= 0) {
        if (Character.isLetter(str.charAt(index))) {
            return index;
        }
    }
    return index;
}
static public boolean palindrom(String str) {
    int i = -1, j = str.length();
    while ((i = nextLetter(str, i)) < (j = previousLetter(str, j))) {
        char leftChar = Character.toLowerCase(str.charAt(i));
        char rightChar = Character.toLowerCase(str.charAt(j));
        if (leftChar != rightChar) {
            return false;
        }
    }
    return true;
}
```

Duplicita

# Čo ak sa chceme alokácii vyhnúť?

```java
static int findLetter(String str, int start, int end, int increment) {
    int index = start;
    while ((index += increment) != end) {
        if (Character.isLetter(str.charAt(index))) {
            return index;
        }
    }
    return index;
}
static int nextLetter(String str, int index) {
    return findLetter(str, index, str.length(), 1);
}
static int previousLetter(String str, int index) {
    return findLetter(str, index, -1, -1);
}

static public boolean palindrom(String str) {
    int i = -1, j = str.length();
    while ((i = nextLetter(str, i)) < (j = previousLetter(str, j))) {
        char leftChar = Character.toLowerCase(str.charAt(i));
        char rightChar = Character.toLowerCase(str.charAt(j));
        if (leftChar != rightChar) {
            return false;
        }
    }
    return true;
}
```

Happy coding …