

آزمایش مهندسی نرم افزار

نیمسال دهم ۱۴۰۰-۱۰

تحویل دهنگان: آرمان محمدی - بردهای خردادری اسلکی

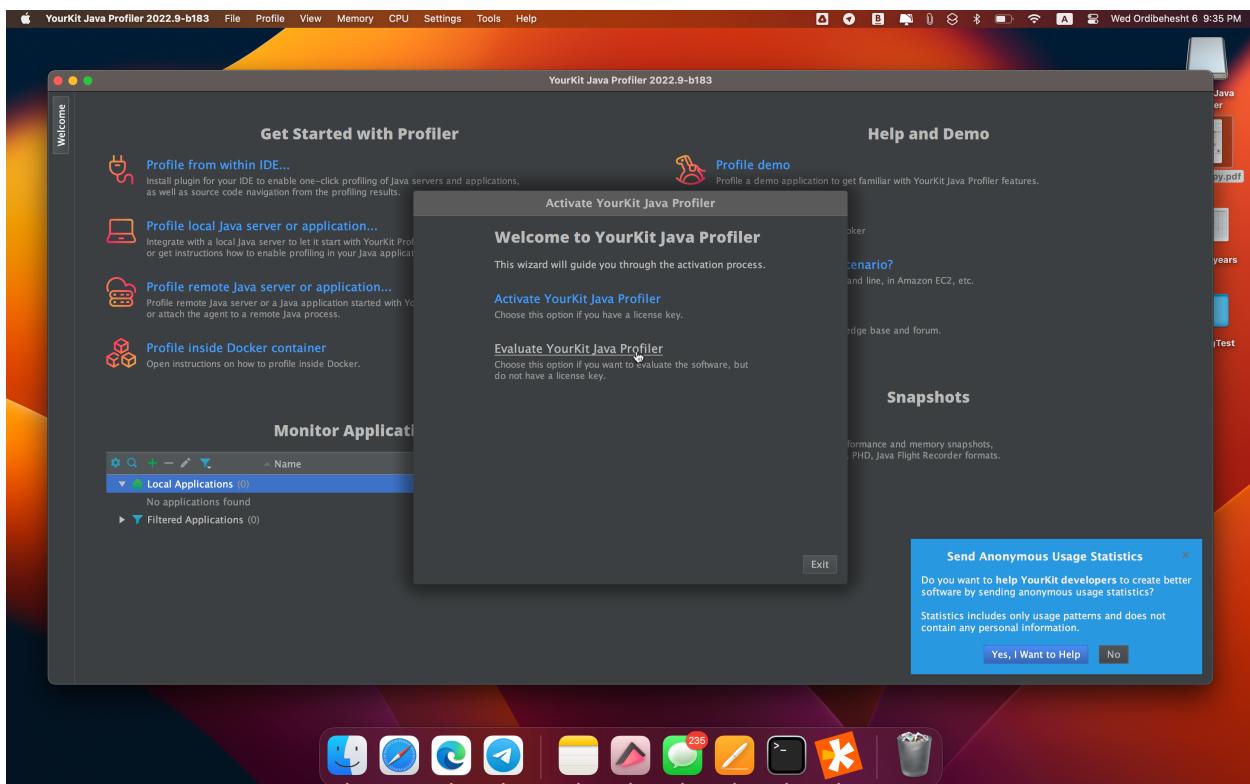


دانشکده مهندسی کامپیوتر

شماره رانشجویی: ۸۱۰۲۳۴۵۶۷۹۰۱۰۰۱۱

آزمایش سری پهارم

ابتدا برنامه YourKit Java را دانلود و نصب می کنیم و روی گزینه Evaluate YourKit Java Profiler کلیک می کنیم:



سپس، ایمیل خود را وارد می کنیم:

The screenshot shows the YourKit website's navigation bar with links for Products, Docs, Forum, Company, and Search. Below the navigation, there's a menu for Java Profiler with links for Overview & Features, Benefits, Download (which is highlighted in blue), Buy & Upgrade, What's New, Early Access, and Docs. At the bottom of the page, there's a large button labeled "Get Evaluation License Key".

Thank you for taking the time to evaluate YourKit Java Profiler.

Your evaluation license key has been sent to rman.mo2000@gmail.com address. You should receive it shortly.

If you do not receive your evaluation license within a minute, please make sure that the email is not blocked by your spam filter or bounced by your email server.

با تأیید ایمیل، کد فعالسازی را دریافت می‌کنیم:

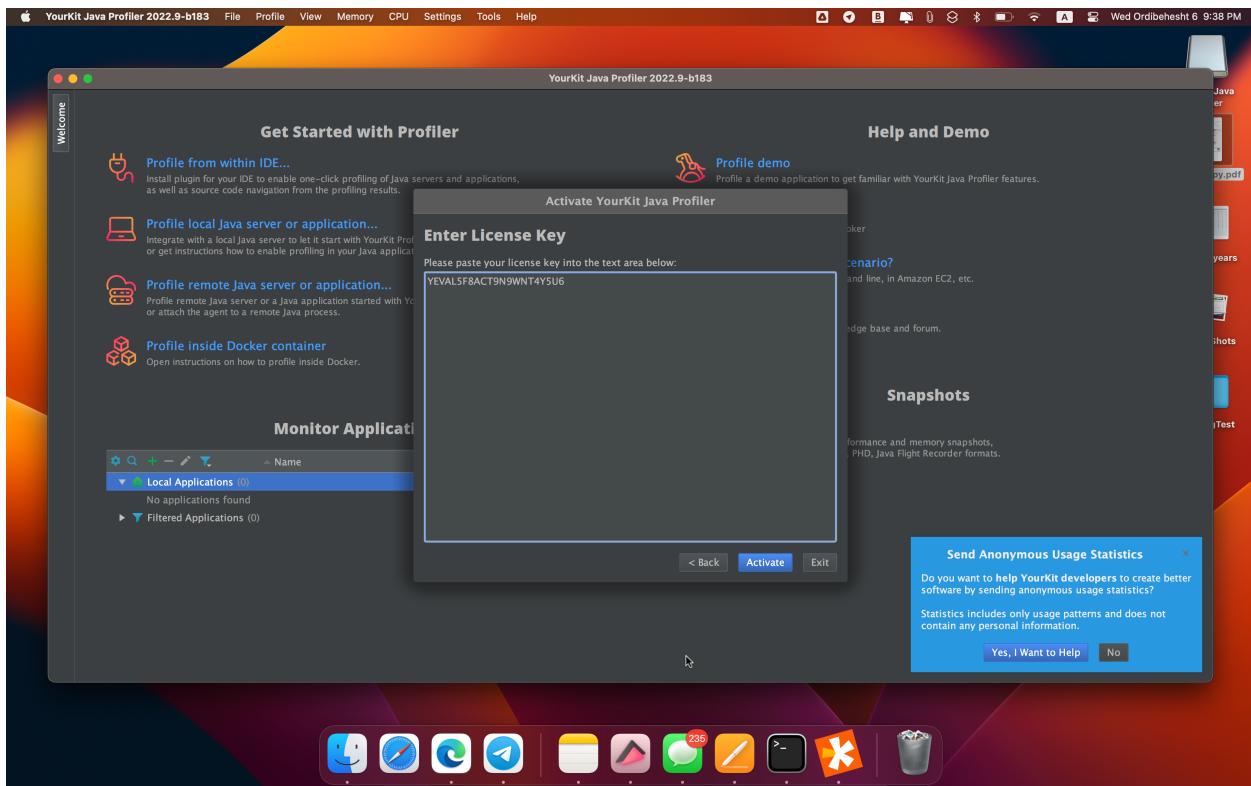
Thank you for trying YourKit Java Profiler!

Your evaluation license key is **YEVAL5F8ACT9N9WNT4Y5U6**. Evaluation period ends on May 11, 2023.

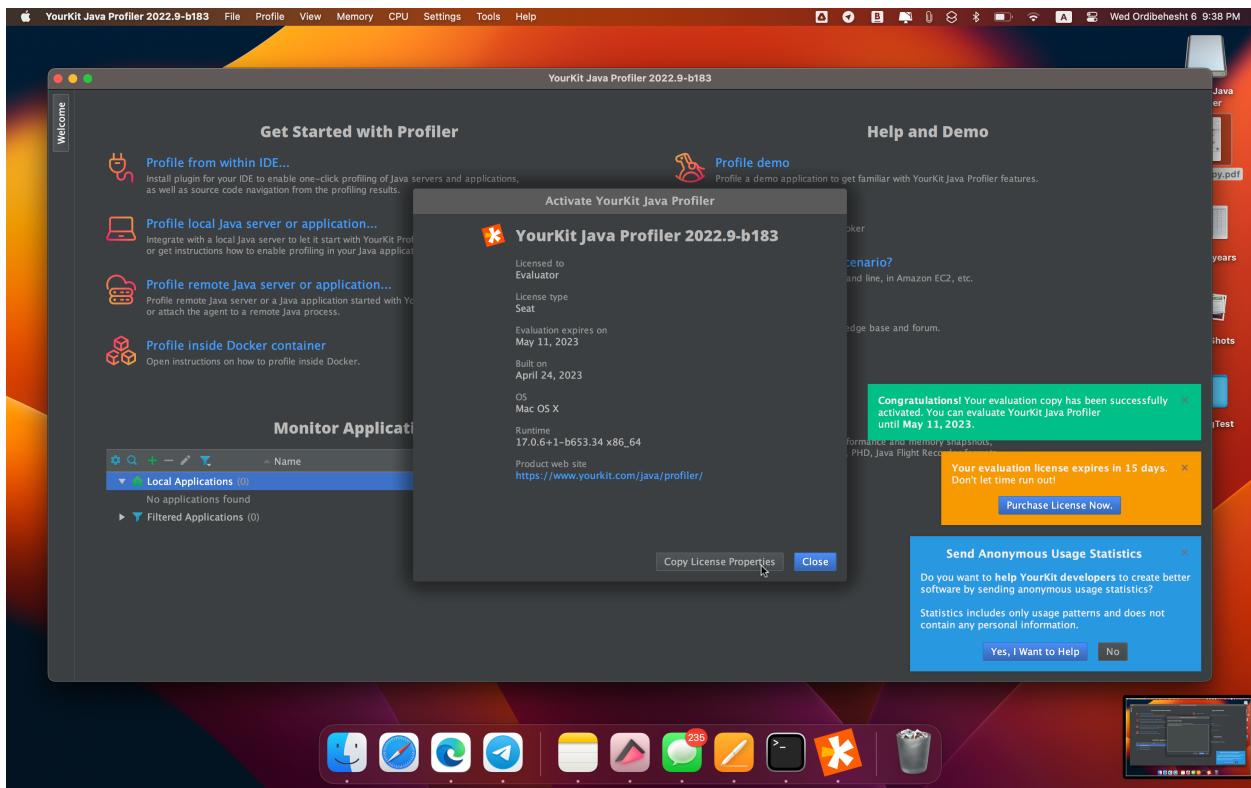
If you have any questions or suggestions during your evaluation period, please do not hesitate to contact us at support@yourkit.com, or ask your questions [in the forum](#).

For your convenience, we have sent evaluation license key on your email address rman.mo2000@gmail.com. You should receive it shortly. If you do not receive our email within a minute, please make sure that the email is not blocked by your spam filter or bounced by your email server.

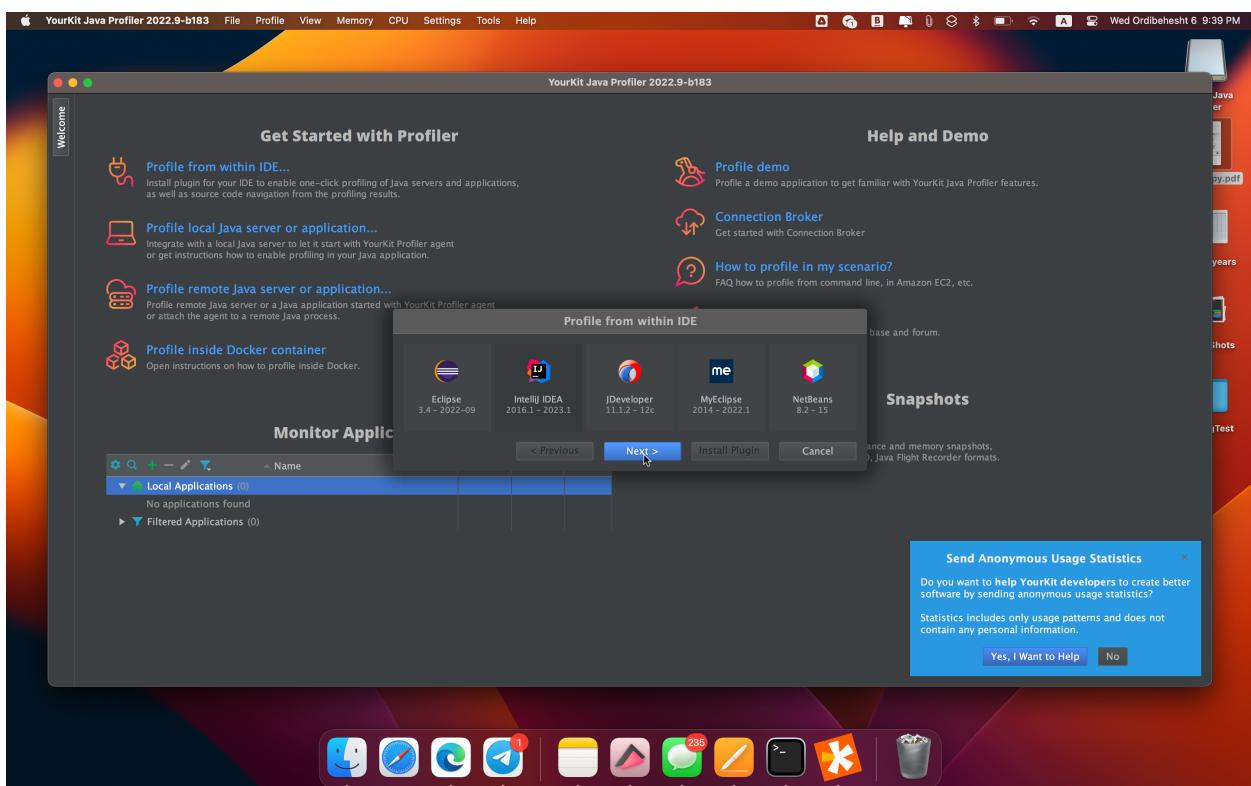
سپس، این کد را در برنامه وارد می‌کنیم:



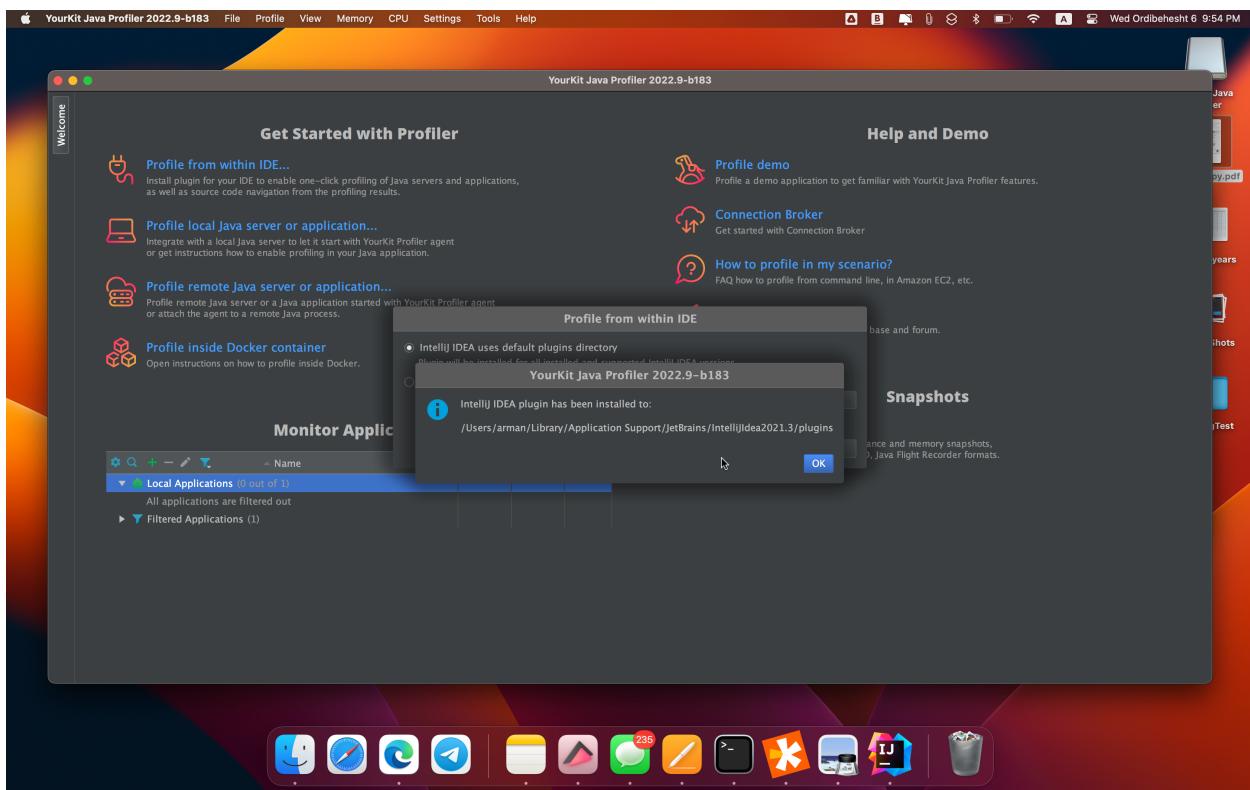
مشاهده می‌کنیم که اکانت ۱۵ روزه‌ی ما فعال شده‌است:



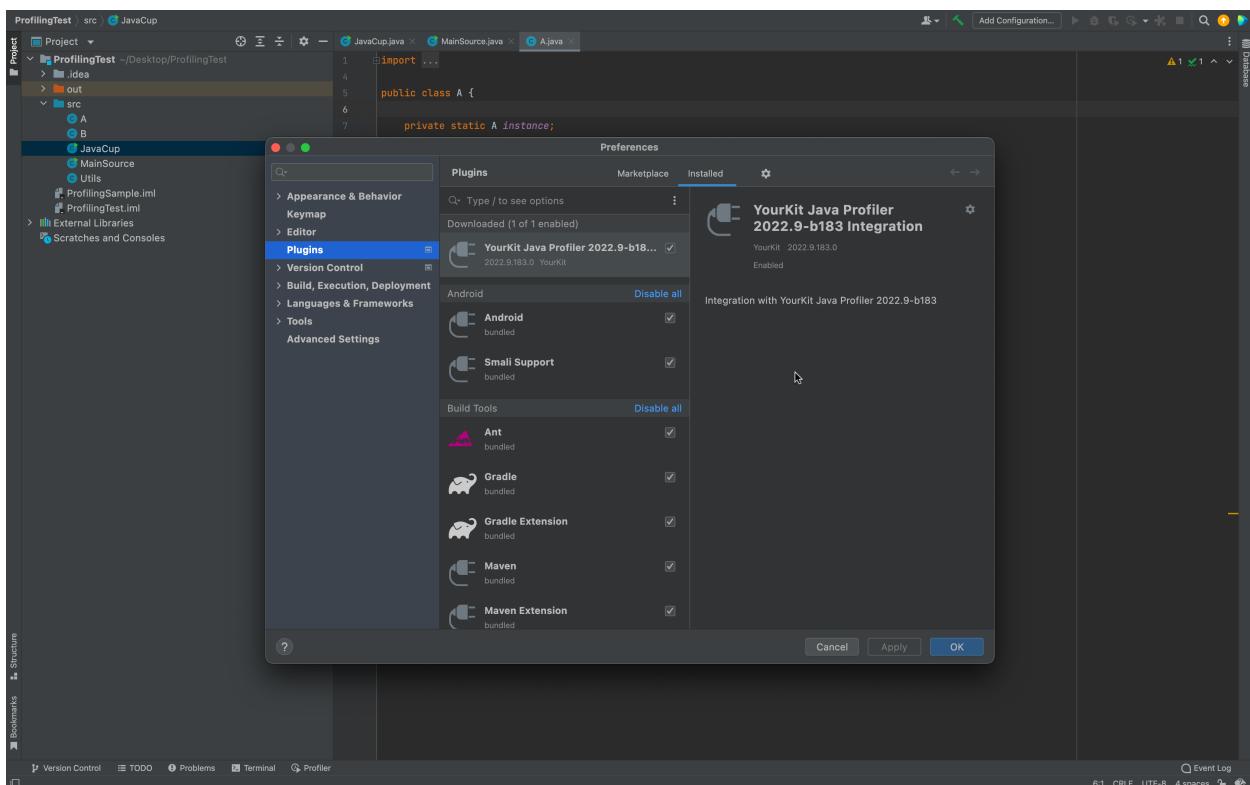
سپس، با کلیک بر روی گزینه‌ی **Profile from within IDE** پلاگین YourKit را روی IntelliJ نصب می‌کنیم:



مشاهده می‌کنیم که عملیات نصب با موفقیت انجام شده است:

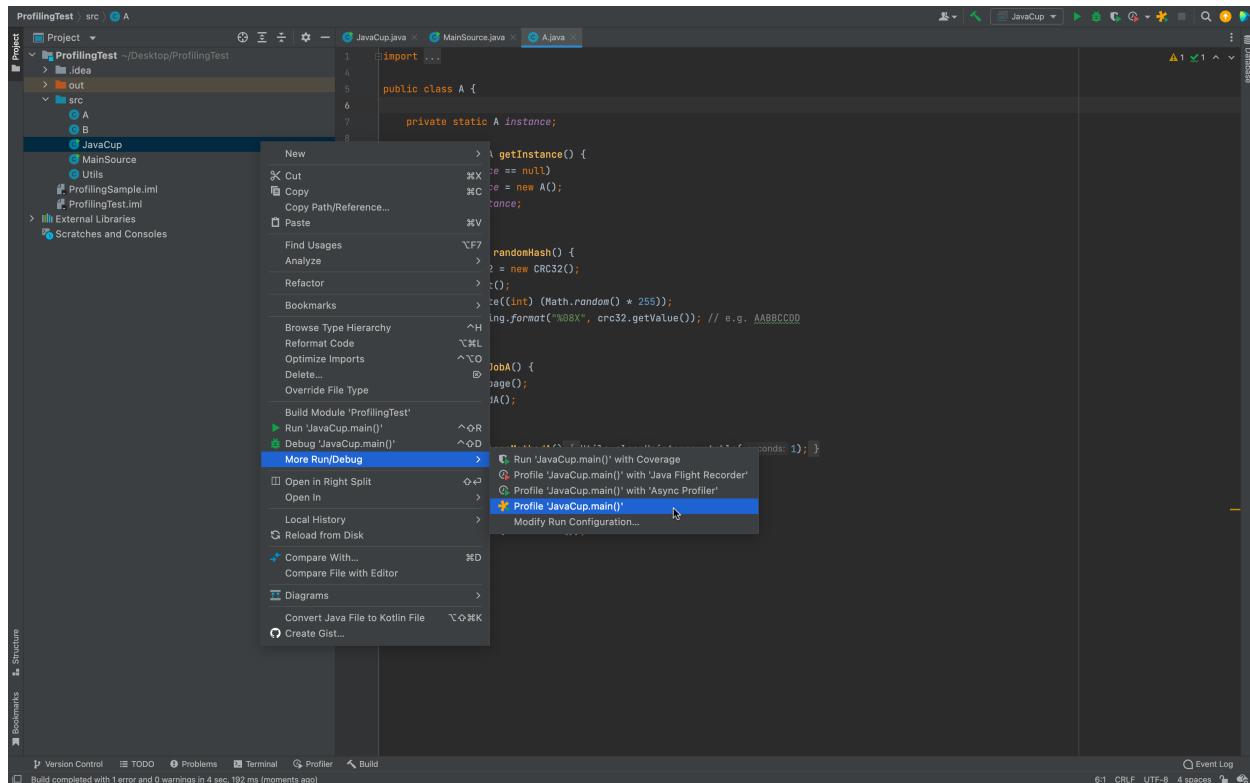


همچنین، با ورود به برنامه‌ی IntelliJ مشاهده می‌کنیم که در پلاگین‌ها مورد نظر در لیست پلاگین‌ها اضافه شده است:

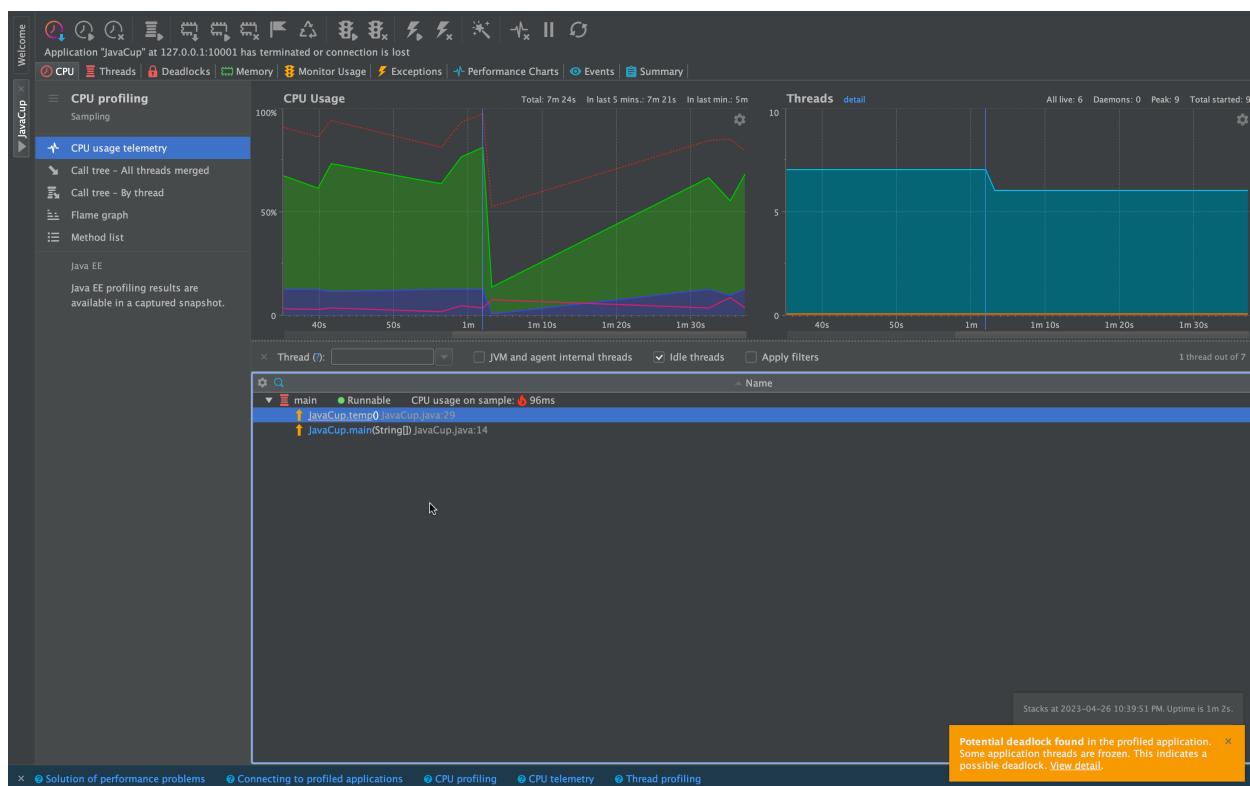


تمرین ۱

حال، عملیات profiling را روی کلاس JavaCup انجام می‌دهیم:



مشاهده می‌کنیم که تابع `temp` بیشترین مصرف منابع را دارد:



با مشاهده کد موجود در کلاس JavaCup، متوجه می‌شویم که در خطوط ۷ تا ۱۳ عملیات خواندن سه عدد انجام می‌شود که در $O(1)$ انجام خواهد شد. همچنین، تابع eval نیز یک سری عملیات جبری و منطقی انجام می‌دهد که در $O(1)$ انجام می‌شوند. ولی همان طور که در خروجی YourKit مشاهده می‌کنیم، خط ۲۹ این کلاس بیشترین مصرف انرژی را دارد و دلیل آن هم این است که دو for تو در تو داریم که سایز اولی ۱۰ هزار و سایز دومی ۲۰ هزار می‌باشد، یعنی در مجموع خط ۲۹ برنامه ۲۰۰ میلیون بار اجرا خواهد شد که به نسبت توان پردازشی و زمان بیشتری نسبت به بقیه قسمت‌های کد نیاز دارد.

```

import ...

public class JavaCup {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Press number1: ");
        int i = scanner.nextInt();
        System.out.println("Press number2: ");
        int j = scanner.nextInt();
        System.out.println("Press number3: ");
        int k = scanner.nextInt();
        temp();
        eval(i, j, k);
    }

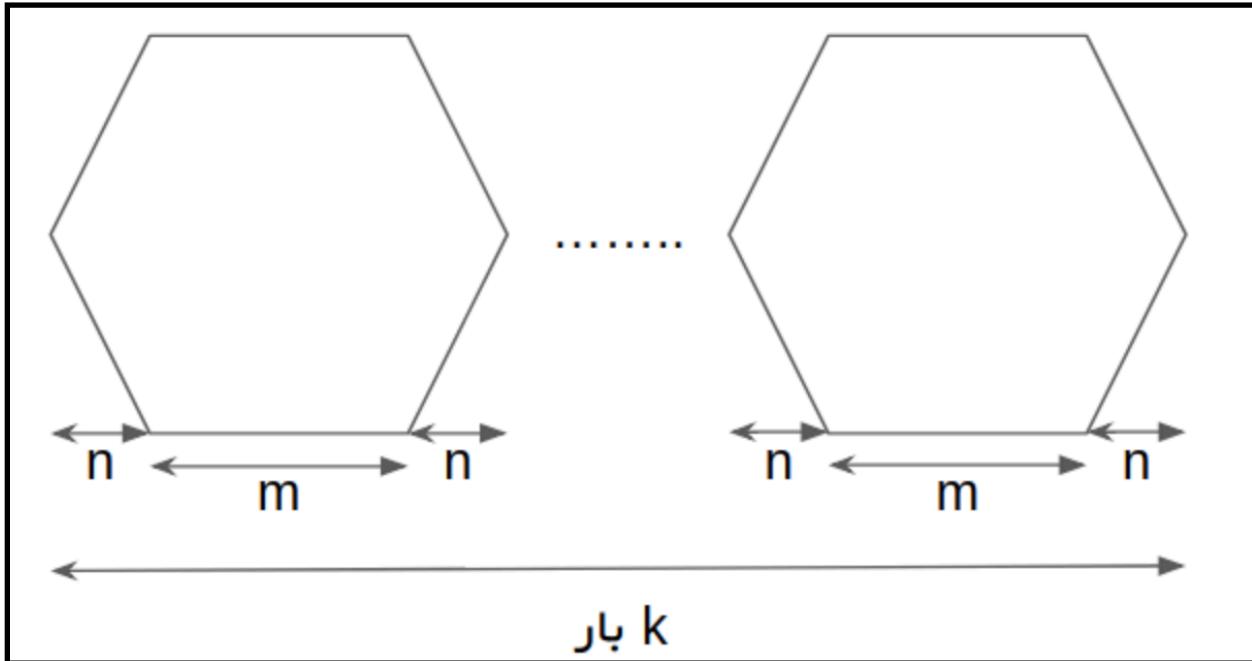
    public static void eval(int i, int j, int k) {
        if (i * i + j * j == k * k || i * i == j * j + k * k || j * j == i * i + k * k)
        {
            System.out.println("YES");
        }
        else { System.out.println("NO"); }
    }

    public static void temp() {
        ArrayList a = new ArrayList();
        for (int i = 0; i < 10000; i++) {
            for (int j = 0; j < 20000; j++) {
                a.add(i + j);
            }
        }
    }
}

```

تمرین ۲

در این قسمت می‌خواهیم کدی بنویسیم که ۶-ضلعی‌هایی همانند شکل زیر چاپ کند:



در واقع این برنامه، ۳ عدد n و m و k را ورودی می‌گیرد و خروجی آن شکل بالا با استفاده از کار *

می‌باشد. برای مثال، برای ورودی‌های

$$n = 2, m = 3, k = 2$$

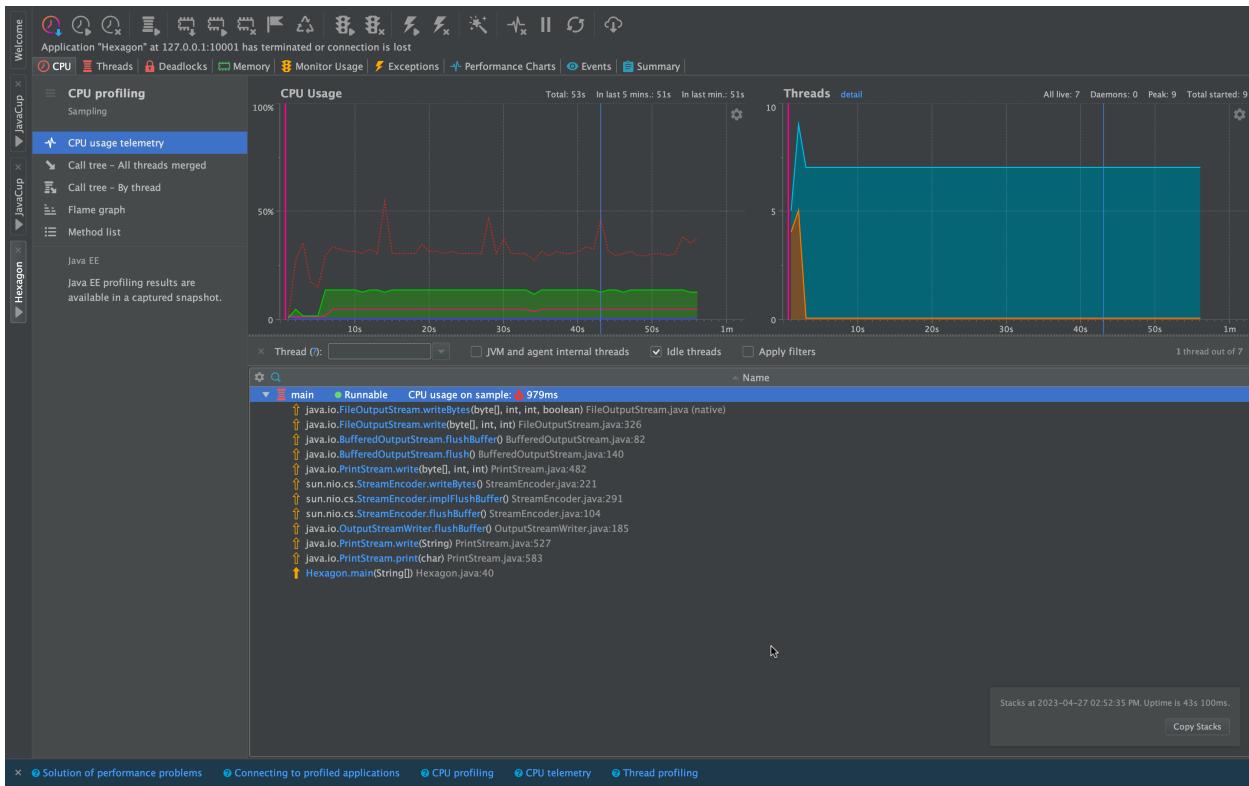
داریم:

```
2 3 2
***      ***
*****  *****
*****
*****  *****
***      ***
```

ابتدا کد برنامه را به صورت زیر تعریف می‌کنیم:

```
1 import java.sql.SQLOutput;
2 import java.util.Scanner;
3
4 public class Hexagon {
5     public static void main(String[] args) {
6         Scanner scanner = new Scanner(System.in);
7         int n = scanner.nextInt();
8         int m = scanner.nextInt();
9         int k = scanner.nextInt();
10        char[][] matrix = new char[2 * n + 1][2 * n + m];
11        int i = n;
12        int row = 0;
13        while (i >= 0) {
14            for (int j = 0; j < 2 * n + m; j++) {
15                if (j < i || 2 * n + m - j <= i)
16                    matrix[row][j] = ' ';
17                else
18                    matrix[row][j] = '*';
19            }
20            i--;
21            row++;
22        }
23        i += 2;
24        while (i <= n) {
25            for (int j = 0; j < 2 * n + m; j++) {
26                if (j < i || 2 * n + m - j <= i)
27                    matrix[row][j] = ' ';
28                else
29                    matrix[row][j] = '*';
30            }
31            i++;
32            row++;
33        }
34        for (int l = 0; l < 2 * n + 1; l++) {
35            for (int a = 1; a <= k; a++) {
36                for (int j = 0; j < 2 * n + m; j++) {
37                    System.out.print(matrix[l][j]);
38                }
39            }
40            System.out.print("\n");
41        }
42    }
43 }
44
45
```

عملیات پروفایل را برای این کد و برای ورودی $n = 100, m = 200, k = 100$ انجام می‌دهیم:



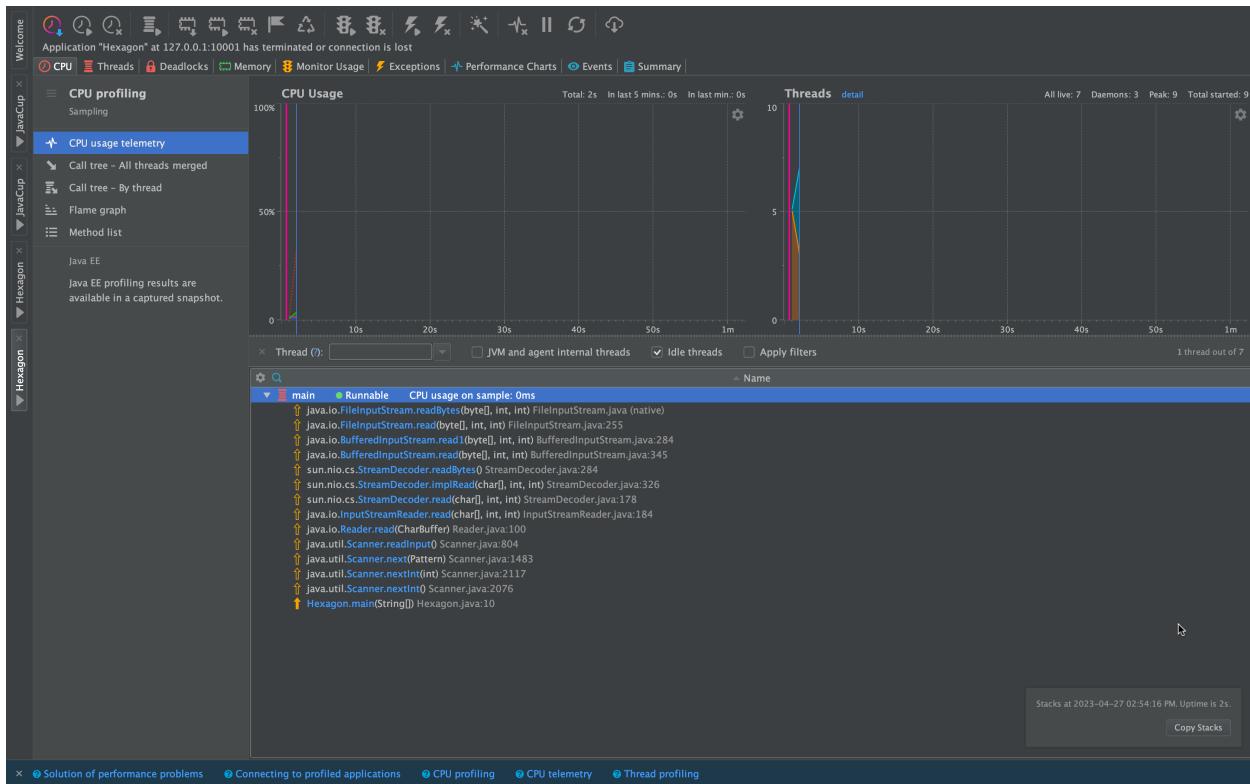
مشاهده می‌کنیم که اجرای برنامه بیش از ۵۰ ثانیه زمان برده است و بیشترین مصرف منابع هم مربوط به خط ۴۰ برنامه بوده است. پس نتیجه می‌گیریم print کردن کارکترها منابع زیادی را مصرف می‌کند و باید عملیات پرینت کردن را سریع‌تر انجام دهیم.

برای این کار، کد بالا را به شکل زیر تغییر می‌دهیم:

```
1 import java.sql.SQLOutput;
2 import java.util.Scanner;
3
4 public class Hexagon {
5     public static void main(String[] args) {
6         Scanner scanner = new Scanner(System.in);
7         int n = scanner.nextInt();
8         int m = scanner.nextInt();
9         int k = scanner.nextInt();
10        String lines;
11        int i = n;
12        while (i >= 0) {
13            lines = "";
14            for (int j = 0; j < 2 * n + m; j++) {
15                if (j < i || 2 * n + m - j <= i)
16                    lines = lines + ' ';
17                else
18                    lines = lines + '*';
19            }
20            for(int a = 1; a <= k; a++)
21                System.out.print(lines);
22            System.out.print("\n");
23            i--;
24        }
25        i += 2;
26        while (i <= n){
27            lines = "";
28            for (int j = 0; j < 2 * n + m; j++) {
29                if (j < i || 2 * n + m - j <= i)
30                    lines = lines + ' ';
31                else
32                    lines = lines + '*';
33            }
34            for(int a = 1; a <= k; a++)
35                System.out.print(lines);
36            System.out.print("\n");
37            i++;
38        }
39    }
40 }
41
42
```

در واقع به جای این که هر کاراکتر را صورت جدا پرینت کنیم، یک رشته از کاراکترهای هر خط تشکیل می‌دهیم و آن رشته را در سیستم پرینت می‌کنیم، یعنی عملیات پرینت کردن به جای $(2n + 1) \times (2n + m)$ بار انجام می‌شود.

حال، عملیات profiling را بر روی این کد بهینه شده اجام می دهیم:



مشاهده می کنیم که زمان اجرای برنامه به طرز چشمگیری کاهش یافته است. همچنین، درصد استفاده از CPU نیز به شدت کاهش یافته است.