SpamCallIdentifer ドキュメント

日本大学 文理学部情報科学科
5419045 高林 秀

2021 年 12 月 27 日

概要

本ドキュメントは、今年度発展プログラミングの課題研究として作成する、Java を使用したマルチスレッド処理を伴うアプリケーションの仕様を説明するものである。

本稿前半部では、本制作の目的、制作したアプリケーションの名前や仕様などの説明、実際に動作させる際の手順を解説する。後半部では、並行処理にまつわる歴史的な背景について軽く説明する。巻末部では本アプリケーションのソースコードを掲載したリポジトリの URL を示しているので、必要に応じてご参照いただきたい。

なお、本アプリケーションは開発環境と同等の環境のみ動作保証対象とする。

1　目的

本稿で紹介するアプリケーションは、今年度発展プログラミングの課題研究として、Java を使用したマルチスレッド処理を伴うアプリケーションを開発することを目的とする。並びに、開発を通して Java、マルチスレッドプログラミングに対する理解を深めることを目的とする。

2　前提知識

本稿では「マルチスレッド」と呼ばれる言葉をしばしば「並行処理」と言い換えて表現する部分がある。

なお、Java の言語特性、基本仕様、アプリケーションに関係のない部分に関する説明は本稿では省略する。

加えて、本アプリケーションの開発にはクラスや、継承などの機能を使用する。その部分に関する説明は以下の URL から、レポート「interface、抽象クラスを利用した Java のペア・プログラミング」を参考いただきたい。

- interface、抽象クラスを利用した Java のペア・プログラミング:`https://drive.google.com/drive/folders/1QEt-NBptDGq2J1BgOyFnGUx8SMwL5oNc?usp=sharing`

3　アプリケーション名

SpamCallIdentifer

3.1　名前の由来

単純に「迷惑電話を判定するもの」という意味で名付けた。

- Spam:「迷惑」という意。
- Call:「電話」。
- Identifer:「判定するもの」という意。

4　アプリケーション概要

SpamCallIdentifer は、入力として電話番号（ハイフン無し）を受け取り、出力としてその電話番号が「スパムである」か「スパムでない」かを判定した結果を表示する。この時、入出力は GUI 表示で行う。

5　想定シーン

発信者が誰かわからない人物から電話がかかって来たとき、そのまま折返し電話をする前に使用することを想定している。

身元不明の人物に折返し電話をするのは非常に危険であり、最悪詐欺などの誘導に引っかかるケースも考えられる。そこで、SpamCallIdentifer に発信者の電話番号を入力し、怪しい発信者か否か判定することで、折

返し電話をするべきか否かの判断の手助けをすることを目的とする。

6　実装上の制約

本アプリケーションは、入力された電話番号をインターネット上で検索し、その番号がスパムに該当するか否か調査する処理の実装を提出環境、開発環境の都合上行っていない。したがって下記のように代替の実装を行っている。

本来の実装	代替の実装
番号の照会処理	Thread.sleep()
照会した電話番号のデータ	JudgeSpam クラス内の配列 spamNumbers

7　クラス・メソッドの説明

今回作成した各クラスを Java ファイルごとに説明する。

- Main.java

メソッド名	説明
main	アプリケーションの呼び出しを行う。 SpamCallIdentifer クラスの display メソッドを呼び出している。

- JudgeSpam.java

メソッド名	説明
run	番号照会を、同クラス内の配列 spamNumbers でシミュレーションする。 spamNumbers 内に入力された番号が含まれていないか判定する。
inputNumber	引数：String 同クラス内のクラス変数 number へ引数の文字列を渡す。

- SpamCallIdentifer.java
  1. SpamCallIdentifer クラス

メソッド名	説明
loadingAnimation	番号照会中のローディングアニメーションのコンポーネントを作成し、返却する。 返り値：JPanel
actionPerformed	描画されたボタンのクリックイベントをリスンし、クリック後の各処理を呼び出す。 引数：ActionEvent
display	コンストラクタで生成した初期コンポーネントの配置と描画を行う。

  2. EndCatcher クラス

メソッド名	説明
run	照会処理（JudgeSpam クラス）のスレッドの終了を監視し、新規ウィンドウを生成し、結果を表示する。

8　並行処理部分の概要

今回は、以下 2 つの部分でマルチスレッドを実装している。

- EndCatcher クラス
- JudgeSpam

マルチスレッドを実装するため、以上の 2 クラスには Runnable インタフェースを implements している。

JudgeSpam クラスは、インターネット上で入力された番号を検索するという、時間のかかる処理を想定している。また此のクラスの run メソッドは、ユーザーが操作する GUI 画面にある「判定開始」ボタンをクリックした時に Thread.start メソッドを介して実行される。

9　並行処理実現のための工夫した点

先程も述べたが、インターネット上で番号検索を行う処理を想定している JudgeSpam クラスは処理に時間のかかるクラスである。したがって、GUI 表示を行う SpamCallIdentifer クラスと同一のスレッドで実行してしまうと、GUI 表示の描画処理が JudgeSpam クラスの run メソッドが終了するまで停止してしまい、画面が応答しなくなってしまう。

通常、GUI アプリケーションでは、画面を制御しているスレッドにて、数十ミリ秒以上時間がかかる処理を実行してはならない。なぜなら、画面の反応が悪く、OS がアプリケーションを応答停止状態と見なし動作を停止してしまうからである。これを回避するため、JudgeSpam クラスの run メソッドを SpamCallIdentifer クラス内のスレッドとは別スレッドで呼び出している。このようにすることで、JudgeSpam クラスの run メソッドの処理と、SpamCallIdentifer クラスの各メソッドが行う GUI 描画更新処理を両方実行し、アプリケーションが応答停止になることを防止している。

10　開発環境

- ホスト OS : Window10 Home 20H2
- 仮想 OS : Ubuntu 20.04.2 LTS
- CPU : Intel(R)Core(TM)i7-9700K @ 3.6GHz
- GPU : Nvidia Geforce RTX2070 OC @ 8GB
- ホスト RAM : 16GB
- 仮想 RAM : 4GB
- 使用エディタ : Microsoft Visual Studio Code
- 使用言語 : Java
  - バージョン情報は下記に示す。

```
openjdk version "11.0.11" 2021-04-20
```

```
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
```

- 利用したライブラリは下記。
  - swing

11 付録：並行処理の必要性

そもそも何故マルチスレッド、マルチプロセスの機能が生まれたのかについて軽く説明する。

初期のコンピュータプログラムは、直接機械語でプログラムが書かれた磁気テープなどを読み取り、直接 CPU に指示を出し実行するものであった。また CPU は基本的に計算装置を 1 つしか持たない「シングルプロセッサ」で 1 つずつ順番にしか処理することのできないものであった。一般に、ハードディスクなどの記憶装置に計算結果を書き込む場合は、CPU での演算にかかる時間に比べ数十〜数百倍の時間がかかるとされる。したがって、1 つ 1 つ順番にしか処理することのできないコンピュータ・プログラムはこれら長い処理が行われている間なにも処理する事ができない「待機時間」が発生することになる。マルチスレッド、マルチプロセスの CPU、プログラムはこれらの待機時間の合間に別の処理を行うことができるので、計算の効率を挙げることができる。

現代では、基本的に OS（オペレーティングシステム）と呼ばれるソフトウェアがそのコンピュータ上で実行される各アプリケーションの制御に関わっている。この OS と呼ばれるプログラムは、アプリケーションの制御だけでなく、次に実行されるプログラムの計算リソースの確保や、ハードウェアの管理などアプリケーションを実行する上で、様々なタスクも同時に行う必要がある。前述したシングルスレッド、シングルプロセスの場合では、2 つ以上のプログラムを同時に実行することはできない。そのためマルチスレッド、マルチプロセスといった並行して処理を行うような仕組みが必要とされた。

12 巻末付録

アプリケーションのソースコードは以下のリポジトリに掲載している。

- GitHub:https://github.com/tsyu12345/advancedPrg