```java
package com.baihaqi.bankingcreditcli;

import java.util.Scanner;

/**
*
* @author G4CE-PC
*         Muhammad Baihaqi Aulia Asy'ari
*         2241720145 - TI 1I - 19
*/
public class BankingCreditCLI {
    final static Scanner input = new Scanner(System.in);
    static String[][] credential = new String[1][2];
    static String username;
    static double[][] creditMortgage = new double[1][4];
    static String[] creditMortgageDetail = new String[1];
    static String[][] profile = new String[1][4];

    public static void main(String[] args) {
        credential[0][0] = "admin";
        credential[0][1] = "admin";
        loginMenu();
    }

    // region login
    static String usernameCheck() {
        while (true) {
            write("Username: ");
            String userInput = input.next();
            for (String[] strings : credential) {
                if (strings[0] == null)
                    continue;
                if (strings[0].equals(userInput))
                    return userInput;
                if (userInput.equalsIgnoreCase("register"))
                    return userInput;
                if (userInput.equalsIgnoreCase("quit"))
                    return userInput;
            }
            printPromptSplit(
                    "The username you've entered doesn't exist in our system please
↪ re-enter your username correctly");
        }
    }

    static boolean passwordCheck() {
        int limit = 0;
        while (limit < 3) {
            write("Password: ");
            String userInput = input.next();
            for (String[] strings : credential) {
                if (strings[0].equals(username)) {
                    if (strings[1].equals(userInput)) {
                        return true;
```

```
                    }
                }
            }
            if (limit < 1)
                writeln("Wrong password");
            if (limit == 1)
                writeln("""
                        Wrong password, Last attempt
                        if you fail again, you would need to re-enter your username""");
            limit++;
        }
        return false;
    }

    static boolean attemptLogin() {
        username = usernameCheck();
        if (!(username.equalsIgnoreCase("register") ||
    ↪ username.equalsIgnoreCase("quit"))) {
            boolean password = passwordCheck();
            if (password) {
                if (profile[getUserID()][0] == null)
                    setProfile();
                mainMenu();
            } else {
                if (attemptLogin()) {
                    if (username.equalsIgnoreCase("register"))
                        registerMenu();
                    if (username.equalsIgnoreCase("quit"))
                        quitMenu();
                }
            }
            return false;
        }
        return true;
    }

    // endregion
    // region print
    static void printHeading(String prompt) {
        int heading;
        String bar;
        String side = "||";
        if (prompt.length() % 2 == 0) {
            heading = 64;
            bar = "================================================================";
        } else {
            heading = 65;
            bar = "=================================================================";
        }
        int gap = ((heading - (side.length() * 2) - prompt.length()) / 2);
        String title = String.format("%s%" + gap + "s%s%" + gap + "s%s\n", side, " ",
    ↪ prompt, " ", side);
        writeln(bar);
        write(title);
```

```
106         writeln(bar);
107     }
108 
109     static void printPromptSplit(String prompt) {
110         String[] promptSplit = prompt.split("\\s");
111         int i = 0;
112         while (i < promptSplit.length) {
113             int limit = 0;
114             while (limit < 65 && i < promptSplit.length && (limit +
    ↪ promptSplit[i].length()) < 65) {
115                 write(String.format("%s ", promptSplit[i]));
116                 limit = limit + (promptSplit[i].length() + 1);
117                 i++;
118             }
119             writeln("");
120         }
121     }
122 
123     static void write(String s) {
124         System.out.print(s);
125     }
126 
127     static void writeln(String s) {
128         System.out.println(s);
129     }
130 
131     // endregion
132     // region etc
133     static void newStringArray(String[][] data) {
134         String[][] old = data;
135         data = new String[old.length + 1][old[0].length];
136         for (int row = 0; row < old.length; row++) {
137             for (int col = 0; col < old[row].length; col++) {
138                 data[row][col] = old[row][col];
139             }
140         }
141     }
142 
143     static void newString(String[] data) {
144         String[] old = data;
145         data = new String[data.length + 1];
146         for (int i = 0; i < old.length; i++) {
147             data[i] = old[i];
148         }
149     }
150 
151     static void newDoubleArray(double[][] data) {
152         double[][] old = data;
153         data = new double[data.length + 1][data[0].length];
154         for (int row = 0; row < old.length; row++) {
155             for (int col = 0; col < old[row].length; col++) {
156                 data[row][col] = old[row][col];
157             }
158         }
```

```
159     }
160
161     static String promptedTextInput(String prompt) {
162         write(prompt);
163         return input.next();
164     }
165
166     static double powerDouble(double base, int exponent) {
167         if (exponent == 0) {
168             return 1;
169         } else {
170             return base * powerDouble(base, exponent - 1);
171         }
172     }
173
174     static void notFound() {
175         writeln("""
176                 _  _     ___  _  _  \s
177                 | || |  / _ \\| || | \s
178                 | || |_| | | | || |_\s
179                 |__   _| | | |__   _|
180                 | | | |_| |  | | \s
181                 |_|  \\___/   |_| \s
182                                 \s
183                                 \s
184                 _  _       _      __                      _\s
185                 | \\| | ___ | |_   / _| ___  _  _  _ _   __| |
186                 | .` |/ _ \\|  _| |  _|/ _ \\| || || ' \\ / _` |
187                 |_|\\_|\\___/ \\__| |_|  \\___/ \\_,_||_||_|\\__,_|
188                                                           \s""");
189     }
190
191     static boolean confirm() {
192         while (true) {
193             write("Are you sure (y/n): ");
194             String userInput = input.next();
195             if (userInput.equalsIgnoreCase("y")) {
196                 return true;
197             } else if (userInput.equalsIgnoreCase("n")) {
198                 return false;
199             }
200             writeln("Please enter a valid input!");
201         }
202     }
203
204     static int getUserID() {
205         int i = 0;
206         if (!credential[i][0].equals(username)) {
207             do {
208                 i++;
209             } while (!credential[i][0].equals(username));
210         }
211         return i;
212     }
```

```
213
214     static void insertMortgageDetail(String s) {
215         write(s + "\n");
216         creditMortgageDetail[getUserID()] += s + "\n";
217     }
218
219     static boolean validatePhoneNumber(String numbers) {
220         // 0895388899808 -> 13 digits
221         // 082336750134 -> 12 digits
222         // 08912888374 -> 11 digits
223         int length = numbers.length();
224
225         if (length < 11 || length > 13)
226             return false;
227
228         for (int i = 0; i < length; i++) {
229             // validate if each element is a number
230
231             int current = Integer.parseInt(String.format("%c", numbers.charAt(i)));
232             // first digit should be 0
233             if (i == 0 && current != 0)
234                 return false;
235
236             // second digit should be 8
237             if (i == 1 && current != 8)
238                 return false;
239         }
240         return true;
241     }
242
243     static boolean validateIDCardNumber(String numbers) {
244         // 3573051004040001
245         // 3573056204040001
246         int length = numbers.length();
247         if (length != 16)
248             return false;
249         for (int i = 0; i < length; i++) {
250             switch (numbers.charAt(i)) {
251                 case '1', '2', '3', '4', '5', '6', '7', '8', '9', '0' -> {
252                 }
253                 default -> {
254                     return false;
255                 }
256             }
257         }
258         return true;
259     }
260
261     static int[] debtReadjustment(double installmentMin, double
    ↪ inverseReturnOfPoweredInterest, double interestInMonth, int downPayment, int
    ↪ downPaymentPercentage) {
262         boolean repeat = true;
263         double creditLimit;
264         double debtMax;
```

```
265         int debt;
266         int[] out = new int[2];
267         do {
268             creditLimit = installmentMin
269                     * (inverseReturnOfPoweredInterest / interestInMonth);
270             debtMax = creditLimit + downPayment;
271             write(String.format("Maximum proposed debt: %,.0f\n", debtMax));
272             write("House price: ");
273             debt = input.nextInt();
274             if (debt > debtMax) {
275                 writeln("Please enter a value smaller than the maximum");
276             } else {
277                 repeat = false;
278             }
279         } while (repeat);
280         if (downPayment < (double) (debt / 100) * downPaymentPercentage) {
281             write(String.format("Minimum down payment amount: %,.0f\n", (double) (debt /
    ↪ 100) * downPaymentPercentage));
282             repeat = true;
283             do {
284                 write("Down payment: ");
285                 downPayment = input.nextInt();
286                 if (downPayment < (debt / 100) * downPaymentPercentage) {
287                     writeln("Please enter a value bigger than the minimum!");
288                 } else {
289                     repeat = false;
290                 }
291             } while (repeat);
292         } else {
293             out[0] = debt;
294             out[1] = downPayment;
295             return out;
296         }
297         if (downPayment > (double) (debt / 100) * downPaymentPercentage) {
298             out = debtReadjustment(installmentMin, inverseReturnOfPoweredInterest,
    ↪ interestInMonth, downPayment,downPaymentPercentage);
299         }
300         out[0] = debt;
301         out[1] = downPayment;
302         return out;
303     }
304 
305     // endregion
306     // region menu
307     static void loginMenu() {
308         printHeading("LOGIN");
309         printPromptSplit(
310                 "If you don't already have an account please type \"register\" in the
    ↪ username input, if you want to quit type \"quit\" in the username input");
311         if (attemptLogin()) {
312             if (username.equalsIgnoreCase("register"))
313                 registerMenu();
314             if (username.equalsIgnoreCase("quit"))
315                 quitMenu();
```

```
316         }
317     }
318 
319     static void registerMenu() {
320         printHeading("REGISTER");
321         newStringArray(credential);
322         newStringArray(profile);
323         newDoubleArray(creditMortgage);
324         newString(creditMortgageDetail);
325         creditMortgageDetail[creditMortgageDetail.length - 1] = "";
326         credential[credential.length - 1][0] = promptedTextInput("Enter your username:
    ↪ ");
327         credential[credential.length - 1][1] = promptedTextInput("Enter your password:
    ↪ ");
328         loginMenu();
329     }
330 
331     static void setProfile() {
332         int id = getUserID();
333         String name, phoneNumber, IDCardNumber, salary;
334         input.nextLine();
335         write("Enter your name: ");
336         name = input.nextLine();
337         boolean i = true;
338         do {
339             write("Enter your phone number: ");
340             phoneNumber = input.next();
341             if (validatePhoneNumber(phoneNumber)) {
342                 i = false;
343             } else {
344                 writeln("Please enter a valid phone number");
345             }
346         } while (i);
347         writeln("Please enter your ID card number in this format");
348         writeln("example: 3573052004691337");
349         i = true;
350         do {
351             write("Enter your ID card number: ");
352             IDCardNumber = input.next();
353             if (validateIDCardNumber(IDCardNumber)) {
354                 i = false;
355             } else {
356                 writeln("Please enter a valid ID card number");
357             }
358         } while (i);
359         salary = promptedTextInput("Enter your salary: ");
360 
361         if (confirm()) {
362             profile[id][0] = name;
363             profile[id][1] = phoneNumber;
364             profile[id][2] = IDCardNumber;
365             profile[id][3] = salary;
366             writeln(profile[id][0]);
367             writeln(profile[id][1]);
```

```
368             writeln(profile[id][2]);
369             writeln(profile[id][3]);
370         } else {
371             setProfile();
372         }
373     }
374 
375     static void mainMenu() {
376         printHeading("MENU");
377         writeln("""
378                 1. Credit card menu
379                 2. Loan menu
380                 3. Account information
381                 4. Log out
382                 5. Quit the program""");
383         switch (promptedTextInput("menu: ")) {
384             case "1" -> creditCardMenu();
385             case "2" -> loanMenu();
386             case "3" -> accountInfoMenu();
387             case "4" -> loginMenu();
388             case "5" -> quitMenu();
389         }
390     }
391 
392     // region mainMenu
393     static void creditCardMenu() {
394         printHeading("CREDIT CARD");
395         writeln("""
396                 1. Apply for a credit card
397                 2. Owned Credit card
398                 3. Back to main menu""");
399         switch (promptedTextInput("menu: ")) {
400             case "1" -> newCreditCard();
401             case "2" -> ownedCreditCard();
402             case "3" -> mainMenu();
403         }
404     }
405 
406     // region creditCardMenu
407     static void newCreditCard() {
408         printHeading("APPLY FOR A CREDIT CARD");
409         writeln("""
410                 1. General purpose
411                 2. Travel
412                 3. Lifestyle
413                 4. Priority
414                 5. Back to credit card menu""");
415         switch (promptedTextInput("menu: ")) {
416             case "1" -> generalPurposeCreditCardApplication();
417             case "2" -> travelCreditCardApplication();
418             case "3" -> lifestyleCreditCardApplication();
419             case "4" -> priorityCreditCardApplication();
420             case "5" -> creditCardMenu();
421         }
```

```
422         }
423
424         // region newCreditCard
425         static void generalPurposeCreditCardApplication() {
426             printHeading("GENERAL PURPOSE CREDIT CARD APPLICATION");
427             notFound();
428             newCreditCard();
429         }
430
431         static void travelCreditCardApplication() {
432             printHeading("TRAVEL CREDIT CARD APPLICATION");
433             notFound();
434             newCreditCard();
435         }
436
437         static void lifestyleCreditCardApplication() {
438             printHeading("LIFESTYLE CREDIT CARD APPLICATION");
439             notFound();
440             newCreditCard();
441         }
442
443         static void priorityCreditCardApplication() {
444             printHeading("PRIORITY CREDIT CARD APPLICATION");
445             notFound();
446             newCreditCard();
447         }
448
449         // endregion
450         static void ownedCreditCard() {
451             printHeading("OWNED CREDIT CARD");
452             notFound();
453             creditCardMenu();
454         }
455
456         // endregion
457         static void loanMenu() {
458             printHeading("LOAN");
459             writeln("""
460                     1. Apply for a loan
461                     2. Current loan status
462                     3. Back to main menu""");
463             switch (promptedTextInput("menu: ")) {
464                 case "1" -> newLoanMenu();
465                 case "2" -> accountLoanInfo();
466                 case "3" -> mainMenu();
467             }
468         }
469
470         // region loanMenu
471         static void newLoanMenu() {
472             printHeading("APPLY FOR A LOAN");
473             writeln("""
474                     1. Personal
475                     2. Auto
```

```
                3. Mortgage
                4. Refinancing
                5. Back to loan menu""");
        switch (promptedTextInput("menu: ")) {
            case "1" -> personalLoanApplication();
            case "2" -> autoLoanApplication();
            case "3" -> mortgageLoanApplication();
            case "4" -> refinancingLoanApplication();
            case "5" -> loanMenu();
        }
    }

    // region newLoanMenu
    static void personalLoanApplication() {
        printHeading("PERSONAL LOAN");
        notFound();
        newLoanMenu();
    }

    static void autoLoanApplication() {
        printHeading("AUTO LOAN");
        notFound();
        newLoanMenu();
    }

    static void mortgageLoanApplication() {
        int id = getUserID();
        int buildingArea;
        int creditFacilities;
        int downPayment;
        int downPaymentPercentage = 10;
        int tenor;
        int debt;
        int salary = Integer.parseInt(profile[id][3]);
        double installment;
        double interest = 7.25;
        double creditLimit;
        double debtMax;
        double installmentMin = salary < 5_000_000 ? salary * 0.5 : salary * 0.55;
        double salaryMin;
        printHeading("MORTGAGE LOAN");
        writeln("""
                Purpose of Credit
                1. Buying a house
                2. Renovating""");
        String menu = promptedTextInput("menu: ");
        if (menu.equals("1")) {
            writeln("""
                    Collateral Type
                    1. House
                    2. Apartment
                    3. Shop""");
            String collateralType = promptedTextInput("menu: ");
            if (collateralType.equals("1") || collateralType.equals("2")) {
```

```
                write("Building Area (m2): ");
                buildingArea = input.nextInt();
                if (buildingArea > 70)
                    downPaymentPercentage += 5;
            }
        } else if (menu.equals("2")) {
            downPaymentPercentage += 20;
        }
        writeln("How many Credit Facilities do you have");
        boolean repeat = true;
        do {
            write("Credit Facility: ");
            creditFacilities = input.nextInt();
            if (creditFacilities < 1) {
                writeln("Please enter a positive value!");
            } else {
                repeat = false;
            }
        } while (repeat);
        if (creditFacilities > 2)
            downPaymentPercentage += 10;
        writeln("Maximum 20 years tenor");
        repeat = true;
        do {
            write("Tenor: ");
            tenor = input.nextInt();
            if (tenor < 1 || tenor > 20) {
                writeln("Please enter a value between 1 to 20");
            } else {
                repeat = false;
            }
        } while (repeat);
        double interestInMonth = ((interest / 100) / 12);
        double interestPowerBase = (1 + ((interest / 100) / 12));
        int tenorMonth = tenor * 12;
        double inverseReturnOfPoweredInterest = 1 - (1 / powerDouble(interestPowerBase,
    ↪  tenorMonth));
        repeat = true;
        do {
            creditLimit = installmentMin
                    * (inverseReturnOfPoweredInterest / interestInMonth);
            debtMax = creditLimit * (1 / ((double) (100 - downPaymentPercentage) / 100));
            write(String.format("Maximum proposed debt: %,.0f\n", debtMax));
            write("House price: ");
            debt = input.nextInt();
            if (debt > debtMax) {
                writeln("Please enter a value smaller than the maximum");
            } else {
                repeat = false;
            }
        } while (repeat);
        write(String.format("Minimum down payment amount: %,.0f\n", (double) (debt / 100)
    ↪  * downPaymentPercentage));
        repeat = true;
```

```
582         do {
583             write("Down payment: ");
584             downPayment = input.nextInt();
585             if (downPayment < (debt / 100) * downPaymentPercentage) {
586                 writeln("Please enter a value bigger than the minimum!");
587             } else {
588                 repeat = false;
589             }
590         } while (repeat);
591         // region to be refactored as a function
592         if (downPayment > (double) (debt / 100) * downPaymentPercentage) {
593             int[] out = debtReadjustment(installmentMin, inverseReturnOfPoweredInterest,
    ↪  interestInMonth, downPayment,downPaymentPercentage);
594             debt = out[0];
595             downPayment = out[1];
596         }
597         // endregion
598         double debtInterest = (debt - downPayment) * interestInMonth;
599 
600         installment = debtInterest / inverseReturnOfPoweredInterest;
601         salaryMin = installment < 2_500_000 ? installment * 2 : installment * (1 / 0.55);
602         write(String.format("%14s IDR %,d\n", "Installment", (long) installment));
603         write(String.format("%14s IDR %,d\n", "Debt principal", (debt - downPayment)));
604         write(String.format("%14s IDR %,d\n", "Minimum Income", (long) salaryMin));
605 
606         String prompt = "Mortgage Application";
607         String ordinal;
608         switch
    ↪  (String.valueOf(creditFacilities).charAt(String.valueOf(creditFacilities).length() -
    ↪  1)) {
609             case '1' -> ordinal = "st";
610             case '2' -> ordinal = "nd";
611             case '3' -> ordinal = "rd";
612             default -> ordinal = "th";
613         }
614         double downPaymentPercentageByDebt = ((double) downPayment / debt) * 100;
615 
616         String[] varValue = {
617                 String.format(": %d%s", creditFacilities, ordinal),
618                 String.format(": IDR %,d", debt),
619                 String.format(": IDR %,d", downPayment),
620                 String.format(": IDR %,d", (debt - downPayment)),
621                 String.format(": %d", tenor),
622                 String.format(": %.2f%s", interest, "%"),
623                 String.format(": IDR %,.2f", installment),
624                 String.format(": IDR %,.2f", salaryMin)
625         };
626 
627         String[] varName = {
628 
629                 "Credit facility ",
630                 "House price ",
631                 String.format("Down payment %.2f%s ", downPaymentPercentageByDebt, "%"),
632                 "Debt principal ",
```

```
633                 "Tenor ",
634                 "Interest ",
635                 "installment ",
636                 "Minimum income "
637         };
638
639         String barTop =
    ↪ "================================================================";
640         String barBot =
    ↪ "||============================================================||";
641         String side = "||";
642         int heading = barTop.length();
643         int headingSpacing = (heading - (2 * side.length()) - prompt.length()) / 2;
644         String contentSpacing = String.format("%s%" + (heading - (2 * side.length())) +
    ↪ "s%s", side, " ", side);
645         String title = String.format("%s%" + headingSpacing + "s%s%" + headingSpacing +
    ↪ "s%s", side, " ", prompt, " ",
646                 side);
647         int fit = 0;
648         for (String varNameElement : varName) {
649             if (fit < varNameElement.length()) {
650                 fit = varNameElement.length();
651             }
652         }
653         insertMortgageDetail(barTop);
654         insertMortgageDetail(title);
655         insertMortgageDetail(barBot);
656         insertMortgageDetail(contentSpacing);
657         for (int i = 0; i < varName.length; i++) {
658             int paddingLeft = ((heading / 2) - side.length() - fit);
659             int paddingRight = ((heading / 2) - side.length() - varValue[i].length());
660             String content = String.format("%s%" + paddingLeft + "s%-" + fit + "s%s%" +
    ↪ paddingRight + "s%s", side,
661                     " ", varName[i], varValue[i], " ", side);
662             insertMortgageDetail(content);
663             insertMortgageDetail(contentSpacing);
664         }
665         insertMortgageDetail(barBot);
666
667         if (confirm()) {
668             creditMortgage[id][0] = debt;
669             creditMortgage[id][1] = tenor;
670             creditMortgage[id][2] = downPayment;
671             creditMortgage[id][3] = installment;
672         }
673         newLoanMenu();
674     }
675
676     static void refinancingLoanApplication() {
677         printHeading("REFINANCING LOAN");
678         notFound();
679         newLoanMenu();
680     }
681
```

```
682     // endregion
683     static void accountLoanInfo() {
684         printHeading("LOAN STATUS");
685         if (creditMortgage[getUserID()][0] == 0) {
686             notFound();
687         } else {
688             write(creditMortgageDetail[getUserID()]);
689         }
690         write("Exit?");
691         if (confirm()) {
692             loanMenu();
693         } else {
694             accountLoanInfo();
695         }
696     }
697 
698     // endregion
699     static void accountInfoMenu() {
700         int id = getUserID();
701         if (profile[id][0] != null) {
702             String content;
703             String prompt = "ACCOUNT INFO";
704             String barTop =
    ↪ "================================================================================";
705             String barBot =
    ↪ "||============================================================================||";
706             String side = "||";
707             int heading = barTop.length();
708             String contentSpacing = String.format("%s%" + (heading - (2 * side.length()))
    ↪ + "s%s", side, " ", side);
709             int titlePadding = (heading - (2 * side.length()) - prompt.length()) / 2;
710 
711             String[] varName = {
712                     "Name ",
713                     "Phone number ",
714                     "ID card number ",
715                     "salary "
716             };
717             String title = String.format("%s%" + titlePadding + "s%s%" + titlePadding +
    ↪ "s%s", side, " ", prompt, " ",
718                     side);
719 
720             writeln(barTop);
721             writeln(title);
722             writeln(barBot);
723             writeln(contentSpacing);
724 
725             int fitVarName = 0;
726             for (String varNameElement : varName) {
727                 if (fitVarName < varNameElement.length()) {
728                     fitVarName = varNameElement.length();
729                 }
730             }
731             int fitVarValue = 0;
```

```
732             for (String varValueElement : profile[id]) {
733                 if (fitVarValue < varValueElement.length()) {
734                     fitVarValue = varValueElement.length();
735                 }
736             }
737 
738             for (int i = 0; i < varName.length; i++) {
739                 String var = String.format("%-" + fitVarName + "s: %-" + fitVarValue +
    ↪  "s", varName[i], profile[id][i]);
740                 int padding = (heading - (side.length() * 2)
741                         - (var.length() % 2 == 0 ? var.length() : (var.length() + 1))) /
    ↪  2;
742                 if (var.length() % 2 != 0) {
743                     content = String.format("%s%" + padding + "s%s%" + padding + "s %s",
    ↪  side, " ", var, " ", side);
744                 } else {
745                     content = String.format("%s%" + padding + "s%s%" + padding + "s%s",
    ↪  side, " ", var, " ", side);
746                 }
747                 writeln(content);
748                 writeln(contentSpacing);
749             }
750             writeln(barBot);
751         } else {
752             printHeading("ACCOUNT INFO");
753             notFound();
754         }
755         write("Exit?\n");
756         if (confirm()) {
757             mainMenu();
758         } else {
759             accountInfoMenu();
760         }
761     }
762 
763     static void quitMenu() {
764         printHeading("QUIT SUCCESSFULLY");
765     }
766     // endregion
767     // endregion
768 }
```