```
class Casino {
	public static void main(String[] args) {
		Player player = new Player("Gambler");
		SlotMachine machine = new SlotMachine((Math.random() < 0.5) ? 5 : 10);
		System.out.printf("Welcome, %s%n", player.getName());
		System.out.printf("You will be using the %s machine today, which costs %d coins.%n",
				(machine.getCost() == 10) ? "original" : "modded", machine.getCost());
		int totalSpins = 0;
		for (int spins = 0; player.getMoney() >= machine.getCost(); totalSpins = ++spins) {
			machine.pull();
			System.out.println(machine);
			player.setMoney(player.getMoney() - machine.getCost() + machine.calculate());
			if (machine.calculate() > 0)
				System.out.printf("HIT!!! You win %d coins!%n", machine.calculate());
			else
				System.out.println("Too bad! No win on this pull.");
			System.out.printf("%s%n%n", player);
		}
		System.out.printf("GAME OVER! You were able to pull %d times before going broke",
				totalSpins);
	}

}

class Player {
	private String name;
	private int money;

	public Player(String name) {
		this.name = name;
		this.money = 100;
	}

	public String getName() {
		return this.name;
	}

	public int getMoney() {
		return this.money;
	}

	public void setMoney(int money) {
		this.money = money;
	}

	public String toString() {
		return String.format("%s has $%d", this.name, this.money);
	}
}

class SlotMachine {
	private int cost, multiplier;
	private SlotReel sr1, sr2, sr3;

	public SlotMachine(int cost) {
		this.cost = cost;
		this.multiplier = cost / 5;
		this.sr1 = new SlotReel();
		this.sr2 = new SlotReel();
		this.sr3 = new SlotReel();
	}

	public void pull() {
		this.sr1.spin();
		this.sr2.spin();
		this.sr3.spin();
	}

	public int getCost() {
		return this.cost;
	}

	private boolean hit3() {
		if (this.sr1.toString().equals("horseshoe")
				|| this.sr1.toString().equals("star")
				|| this.sr2.toString().equals("horseshoe")
				|| this.sr2.toString().equals("star")
				|| this.sr3.toString().equals("horseshoe")
				|| this.sr3.toString().equals("star"))
			return false;
		if (this.sr1.equals(sr2) && this.sr2.equals(sr3))
```

```
82                          return true;
83                  return false;
84          }
85
86          private boolean hit2() {
87                  if (this.sr1.toString().equals("horseshoe") && this.sr2.toString().equals("horseshoe"))
88                          return true;
89                  return false;
90          }
91
92          public int calculate() {
93                  if (hit3()) {
94                          switch (this.sr1.toString()) {
95                                  case "spade":
96                                          return this.multiplier * 20;
97                                  case "diamond":
98                                          return this.multiplier * 30;
99                                  case "heart":
100                                         return this.multiplier * 40;
101                                 case "Liberty Bell":
102                                         return this.multiplier * 50;
103                                 default:
104                                         break;
105                         }
106                 }
107                 if (hit2() && this.sr3.toString().equals("star"))
108                         return this.multiplier * 10;
109                 if (hit2())
110                         return this.multiplier * 5;
111                 return 0;
112         }
113
114         public String toString() {
115                 return String.format("%d: %s|%s|%s", this.cost, this.sr1.toString(), this.sr2.toString(),
116                                 this.sr3.toString());
117         }
118 }
119
120 class SlotReel {
121         private int value;
122
123         public void spin() {
124                 this.value = (int) (Math.random() * 6) + 1;
125         }
126
127         public boolean equals(SlotReel reel) {
128                 return this.value == reel.getValue();
129         }
130
131         public int getValue() {
132                 return this.value;
133         }
134
135         public String toString() {
136                 switch (this.value) {
137                         case 1:
138                                 return "diamond";
139                         case 2:
140                                 return "heart";
141                         case 3:
142                                 return "spade";
143                         case 4:
144                                 return "horseshoe";
145                         case 5:
146                                 return "star";
147                         case 6:
148                                 return "Liberty Bell";
149                         default:
150                                 return "";
151                 }
152         }
153 }
154
155 /* This program calculates the winning baseball team
156  * Baseball:
157  * Based on input of 2 baseball team names and 3 baseball players per team (6 total)
158  * Statistics:
159  *      First baseman can score 10 runs exclusive
160  *      Second baseman can score 15 runs exclusive
161  *      Third baseman can score 20 runs exclusive
162  * The runs scored by each player are totaled up into the full score
```

```
 */

/* Imports the scanner module */
import java.util.Scanner; /* dd */

public class FeigenbaumBaseball
{
        public static void main(String[] args)
        {
                // Tells the client about the program
                System.out.print("This program outputs fantasy baseball satistics\n");
                System.out.print("The score of a team is calculated based on the scores of the three
inputted players.\n");
                System.out.print("The scores of the three players are totaled up to create the overall
team score.\n");
                System.out.print("The team with the highest total score wins the simulation.\n\n");

                //Opens Scanner to collect client input
                Scanner scanString = new Scanner(System.in);
                Scanner scanInt = new Scanner(System.in);

                // Collects user input for teamA
                System.out.print("What is the name of your baseball team?\n=> ");
                String teamA = scanString.nextLine();
                System.out.print("What is the name of the first baseman?\n=> ");
                String playerA1 = scanString.nextLine();
                System.out.print("What is the name of the second baseman?\n=> ");
                String playerA2 = scanString.nextLine();
                System.out.print("What is the name of the third baseman?\n=> ");
                String playerA3 = scanString.nextLine();

                // Declares player score variables for team A
                int playerA1Score = (int) (Math.random() * 10) ;
                int playerA2Score = (int) (Math.random() * 15);
                int playerA3Score = (int) (Math.random() * 20);
                int teamAScore = playerA1Score + playerA2Score + playerA3Score;

                // Prints team statistics to standard output
                System.out.printf("\nStatistics for the %s:\n", teamA);
                System.out.printf("%s (first baseman): %s runs\n", playerA1, playerA1Score);
                System.out.printf("%s (second baseman): %s runs\n", playerA2, playerA2Score);
                System.out.printf("%s (third baseman): %s runs\n", playerA3, playerA3Score);
                System.out.printf("the %s has %s runs in total.\n\n", teamA, teamAScore);

                // Collects user input for teamB
                System.out.printf("What team would you like to compare to the %s?\n=> ", teamA);
                String teamB = scanString.nextLine();
                System.out.print("What is the name of the first baseman?\n=> ");
                String playerB1 = scanString.nextLine();
                System.out.print("What is the name of the second baseman?\n=> ");
                String playerB2 = scanString.nextLine();
                System.out.print("What is the name of the third baseman?\n=> ");
                String playerB3 = scanString.nextLine();

                // Closes scanners to prevent resource leak
                scanString.close();
                scanInt.close();

                // Declares player score variables for team A
                int playerB1Score = (int) (Math.random() * 10);
                int playerB2Score = (int) (Math.random() * 15);
                int playerB3Score = (int) (Math.random() * 20);
                int teamBScore = playerB1Score + playerB2Score + playerB3Score;

                // Prints team statistics to standard output
                System.out.printf("\nStatistics for the %s:\n", teamB);
                System.out.printf("%s (first baseman): %s runs\n", playerB1, playerB1Score);
                System.out.printf("%s (second baseman): %s runs\n", playerB2, playerB2Score);
                System.out.printf("%s (third baseman): %s runs\n", playerB3, playerB3Score);
                System.out.printf("the %s has %s runs in total.\n\n", teamB, teamBScore);

                // Defines the graphics as variables
                String teamAGraphic =
                        "     ---\n" +
                        "   =======\n" +
                        "   | X X |\n" +
                        "   |  /  |\n" +
                        "   |(---)|\n" +
                        "   |-----|\n" +
                        "---|     |---\n" +
                        "   -------\n";
```

```
242
243                 String teamBGraphic =
244                         "     ())\n" +
245                         "   (((()))\n" +
246                         "     ())\n" +
247                         "     |  |\n" +
248                         "  ---|  |\n" +
249                         "     |  |----\n" +
250                         "=====|  |\n" +
251                         "     |  |\n";
252
253                 String neutralGraphic =
254                         "o:;::loc::c:..\n" +
255                         "xo:x0000Oo:::;'\n" +
256                         ";;d0KK0O0Kx::::\n" +
257                         "' d0XXX00KXc:::\n" +
258                         ". c000OOOOo::::\n" +
259                         "...c000000;.;::\n" +
260                         " .;l0000o...,::\n" +
261                         ".::oooxkxo:;;:::\n" +
262                         "lllllooooolc':''\n";
263
264                 // Conditional to decide who won the contest
265                 int scoreWonDifference; // Variable shows how much the winning team won by
266                 if(teamAScore > teamBScore) // Scenario in which teamA wins
267                 {
268                         scoreWonDifference = teamAScore - teamBScore;
269                         System.out.printf("The %s won by %s runs!\n\n", teamA, scoreWonDifference);
270                         System.out.print(teamAGraphic);
271                 }
272                 else if(teamAScore == teamBScore) // Scenario in which there is a tie
273                 {
274                         System.out.print("I have no strong feelings one way or the other.\n\n");
275                         System.out.print(neutralGraphic);
276                 }
277                 else // Scenario in which teamB wins
278                 {
279                         scoreWonDifference = teamBScore - teamAScore;
280                         System.out.printf("The %s won by %s runs!\n\n", teamB, scoreWonDifference);
281                         System.out.print(teamBGraphic);
282                 }
283         }
284 }
285 /* This program calculates the winning soccer team
286  * Baseball:
287  * Based on input of 2 soccer team names and 3 soccer players per team (6 total)
288  * Statistics:
289  *      Striker can score 3 goals exclusive
290  *      Left winger can score 5 goals exclusive
291  *      Right winger can score 7 goals exclusive
292  * The goals scored by each player are totaled up into the full score
293  */
294
295 import java.util.Scanner; // Imports the Scanner module
296
297 public class FeigenbaumSoccer
298 {
299         public static void main(String[] args)
300         {
301                 // Tells the client about the program
302                 System.out.print("This program outputs fantasy soccer satistics\n");
303                 System.out.print("The score of a team is calculated based on the scores of the three
inputted players.\n");
304                 System.out.print("The scores of the three players are totaled up to create the overall
team score.\n");
305                 System.out.print("The team with the highest total score wins the simulation.\n\n");
306
307                 //Opens Scanner to collect client input
308                 Scanner scanString = new Scanner(System.in);
309                 Scanner scanInt = new Scanner(System.in);
310
311                 // Collects user input for teamA
312                 System.out.print("What is the name of your soccer team?\n=> ");
313                 String teamA = scanString.nextLine();
314                 System.out.print("What is the name of the striker?\n=> ");
315                 String playerA1 = scanString.nextLine();
316                 System.out.print("What is the name of the left winger?\n=> ");
317                 String playerA2 = scanString.nextLine();
318                 System.out.print("What is the name of the right winger?\n=> ");
319                 String playerA3 = scanString.nextLine();
320
```

```
                // Declares player score variables for team A
                int playerA1Score = (int) (Math.random() * 3) ;
                int playerA2Score = (int) (Math.random() * 5);
                int playerA3Score = (int) (Math.random() * 7);
                int teamAScore = playerA1Score + playerA2Score + playerA3Score;

                // Prints team statistics to standard output
                System.out.printf("\nStatistics for the %s:\n", teamA);
                System.out.printf("%s (striker): %s goals\n", playerA1, playerA1Score);
                System.out.printf("%s (left winger): %s goals\n", playerA2, playerA2Score);
                System.out.printf("%s (right winger): %s goals\n", playerA3, playerA3Score);
                System.out.printf("the %s has %s goals in total.\n\n", teamA, teamAScore);

                // Collects user input for teamB
                System.out.printf("What team would you like to compare to the %s?\n=> ", teamA);
                String teamB = scanString.nextLine();
                System.out.print("What is the name of the striker?\n=> ");
                String playerB1 = scanString.nextLine();
                System.out.print("What is the name of the left winger?\n=> ");
                String playerB2 = scanString.nextLine();
                System.out.print("What is the name of the right winger?\n=> ");
                String playerB3 = scanString.nextLine();

                // Closes scanners to prevent resource leak
                scanString.close();
                scanInt.close();

                        // Declares player score variables for team A
                int playerB1Score = (int) (Math.random() * 3);
                int playerB2Score = (int) (Math.random() * 5);
                int playerB3Score = (int) (Math.random() * 7);
                int teamBScore = playerB1Score + playerB2Score + playerB3Score;

                // Prints team statistics to standard output
                System.out.printf("\nStatistics for the %s:\n", teamB);
                System.out.printf("%s (striker): %s goals\n", playerB1, playerB1Score);
                System.out.printf("%s (left winger): %s goals\n", playerB2, playerB2Score);
                System.out.printf("%s (right winger): %s goals\n", playerB3, playerB3Score);
                System.out.printf("the %s has %s goals in total.\n\n", teamB, teamBScore);

                // Defines the graphics as variables
                String teamAGraphic =
                        "     ---\n" +
                        "   =======\n" +
                        "   | X X |\n" +
                        "   |  /  |\n" +
                        "   |(---)|\n" +
                        "   |-----|\n" +
                        "---|     |---\n" +
                        "   -------\n";

                String teamBGraphic =
                        "     (())\n" +
                        "   (((())))\n" +
                        "     (())\n" +
                        "     |  |\n" +
                        "  ---|  |\n" +
                        "     |  |----\n" +
                        "=====|  |\n" +
                        "     |  |\n";

                String neutralGraphic =
                        "o:;::loc::c:..\n" +
                        "xo:x00000o:::;'\n" +
                        ";;d0KK000Kx::::\n" +
                        "' d0XXX00KXc:::\n" +
                        ". c000OOOOo::::\n" +
                        "...c0OOOO0;.;::\n" +
                        " .;l0000o...,::\n" +
                        ".:oooxkxo:;;:::\n" +
                        "lllllooooolc':'\n";

                // Conditional to decide who won the contest
                int scoreWonDifference; // Variable shows how much the winning team won by
                if(teamAScore > teamBScore) // Scenario in which teamA wins
                {
                        scoreWonDifference = teamAScore - teamBScore;
                        System.out.printf("The %s won by %s goals!\n\n", teamA, scoreWonDifference);
                        System.out.print(teamAGraphic);
                }
                else if(teamAScore == teamBScore) // Scenario in which there is a tie
```

```
402                 {
403                         System.out.print("I have no strong feelings one way or the other.\n\n");
404                         System.out.print(neutralGraphic);
405                 }
406                 else // Scenario in which teamB wins
407                 {
408                         scoreWonDifference = teamBScore - teamAScore;
409                         System.out.printf("The %s won by %s goals!\n\n", teamB, scoreWonDifference);
410                         System.out.print(teamBGraphic);
411                 }
412         }
413 }
414 /** This program is designed to simulate a cashier
415  * The program executes the following steps:
416  *      1. Greets the user with a description of the program
417  *      2. Opens 2 scanners for numerical and string input
418  *      3. Prompts the user to enter how many items they
419  *      bought which is used to define a for loop
420  *      4. A for loop loops for the number of items
421  *      and prompts the user to enter the name and
422  *      price of each item which are added to
423  *      itemListFormatted and moneyTotal
424  *      5. The tax is determined by multiplying the money
425  *      by 6% (the tax rate)
426  *      6. The tax is added to moneyTotal to create
427  *      moneyTotalWithTax
428  *      7. The receipt is printed to the user with their total
429  *      8. The user is asked to enter their payment
430  *      is asked to enter a satisfactory payment with a while loop
431  *      9. Change is calculated into variables for their respective unit
432  *      of currency
433  *      10. The amount of change owed is presented to the user
434  */
435
436 /* Imports scanner module */
437 import java.util.Scanner;
438
439 public class FeigenbaumU2 {
440         public static void main(String[] args) {
441                 /* Displays program instructions (welcome message) */
442                 System.out.print("Welcome to the magnificent Java powered store.\n"
443                                 + "This program takes your purchased items as input and will output your
total price (with tax).\n"
444                                 + "The program will then prompt you to give your money to the
register.\n"
445                                 + "The Java Cashier will then return you your change and confirm your
purchase went successfully.\n");
446
447                 /* Opens scanner for string input */
448                 Scanner scanString = new Scanner(System.in);
449                 /* Opens scanner for numerical input */
450                 Scanner scanNum = new Scanner(System.in);
451
452                 /* Prompts user to enter how many items they plan on checking out */
453                 System.out.print("How many items do you plan on checking out? \n" + "=> ");
454                 /* Collects user input for number of items */
455                 int numberOfItems = scanNum.nextInt();
456
457                 /* Creates string to format item names and prices */
458                 String itemListFormatted = "\nYou purchased:\n";
459                 /* Creates integer for item prices in pennies */
460                 int moneyTotal = 0;
461                 /* Creates string to store current input */
462                 String tempItemName;
463                 /* Creates double to store current item price */
464                 double tempItemPrice;
465                 /* For loop for user to enter all their items */
466                 for (int itemNumber = 0; itemNumber < numberOfItems; itemNumber++) {
467                         /* Prompts user each time they enter an item */
468                         System.out.print("Please enter the name of your item\n" + "=> ");
469                         /* Adds item name to tempItemName */
470                         tempItemName = scanString.nextLine();
471                         /* Prompts user to enter price of that item */
472                         System.out.print("Please enter the price of your item\n" + "=> ");
473                         /* Adds item price to tempItemPrice */
474                         tempItemPrice = scanNum.nextDouble();
475                         /* Adds item and price to itemListFormatted */
476                         itemListFormatted += String.format("%s: $%.2f\n", tempItemName, tempItemPrice);
477                         /* Adds item price to moneyTotal */
478                         moneyTotal += (int) (tempItemPrice * 100);
479                 }
```

```
 480
 481                 /* Computates tax in pennies */
 482                 double taxrate = 0.06;
 483                 int tax = (int) (moneyTotal * taxrate);
 484                 int moneyTotalWithTax = moneyTotal + tax;
 485
 486                 /* Adds subtotal, tax and total to itemListFormatted */
 487                 itemListFormatted += String.format("Subtotal: $%.2f\n" + "Tax: $%.2f\n" + "Total:
$%.2f\n\n",
 488                                 ((double) (moneyTotal)) / 100, ((double) (tax)) / 100, ((double)
(moneyTotalWithTax)) / 100);
 489
 490                 /* Prints out purchased items with prices */
 491                 System.out.print(itemListFormatted);
 492
 493                 /*
 494                  * Asks user to submit payment and collects that payment in pennies into
 495                  * variable "payment"
 496                  */
 497                 int payment = 0;
 498                 System.out.println("Please submit your payment");
 499
 500                 /* Loops until user submits payment equal to or above moneyTotalWithTax */
 501                 do {
 502                         System.out.print("=> ");
 503                         payment = (int) (scanNum.nextDouble() * 100);
 504                         if (payment < moneyTotalWithTax)
 505                                 System.out.printf("You still owe $%.2f%n", ((double)  (moneyTotalWithTax
- payment)) / 100);
 506                 } while (payment < moneyTotalWithTax);
 507
 508                 /* Determine how much change of each type is required */
 509                 int totalChange = payment - moneyTotalWithTax;
 510                 int changeLeft = totalChange;
 511                 int dollars = changeLeft / 100;
 512                 changeLeft %= 100;
 513                 int quarters = changeLeft / 25;
 514                 changeLeft %= 25;
 515                 int dimes = changeLeft / 10;
 516                 changeLeft %= 10;
 517                 int nickels = changeLeft / 5;
 518                 changeLeft %= 5;
 519                 int pennies = changeLeft;
 520
 521                 /*
 522                  * Outputs amount of change owed and sections it into dollars, quarters, dimes,
 523                  * nickels, and pennies
 524                  */
 525                 System.out.printf("Thank you for your payment!\n"
 526                                 + "Your change owed is $%.2f.\n"
 527                                 + "Your change is %d dollars, %d quarters, %d dimes, %d nickels, and %d
pennies.",
 528                                 ((double) (totalChange)) / 100, dollars, quarters, dimes, nickels,
pennies);
 529         }
 530 }
 531 /** This program is designed to simulate a cashier
 532  * The program executes the following steps:
 533  *      1. Greets the user with a description of the program
 534  *      2. Opens 2 scanners for numerical and string input
 535  *      3. Prompts the user to enter how many items they
 536  *      bought which is used to define a for loop
 537  *      4. A for loop loops for the number of items
 538  *      and prompts the user to enter the name and
 539  *      price of each item which are added to
 540  *      itemListFormatted and moneyTotal
 541  *      5. The tax is determined by multiplying the money
 542  *      by 6% (the tax rate)
 543  *      6. The tax is added to moneyTotal to create
 544  *      moneyTotalWithTax
 545  *      7. The receipt is printed to the user with their total
 546  *      8. The user is asked to enter their payment
 547  *      is asked to enter a satisfactory payment with a while loop
 548  *      9. Change is calculated into variables for their respective unit
 549  *      of currency
 550  *      10. The amount of change owed is presented to the user
 551  */
 552
 553 /* Imports scanner module */
 554 import java.util.Scanner;
 555
```

```
556 public class FeigenbaumU2 {
557         public static void main(String[] args) {
558                 /* Displays program instructions (welcome message) */
559                 System.out.print("Welcome to the magnificent Java powered store.\n"
560                                 + "This program takes your purchased items as input and will output your
total price (with tax).\n"
561                                 + "The program will then prompt you to give your money to the
register.\n"
562                                 + "The Java Cashier will then return you your change and confirm your
purchase went successfully.\n");
563
564                 /* Opens scanner for string input */
565                 Scanner scanString = new Scanner(System.in);
566                 /* Opens scanner for numerical input */
567                 Scanner scanNum = new Scanner(System.in);
568
569                 /* Prompts user to enter how many items they plan on checking out */
570                 System.out.print("How many items do you plan on checking out? \n" + "=> ");
571                 /* Collects user input for number of items */
572                 int numberOfItems = scanNum.nextInt();
573
574                 /* Creates string to format item names and prices */
575                 String itemListFormatted = "\nYou purchased:\n";
576                 /* Creates integer for item prices in pennies */
577                 int moneyTotal = 0;
578                 /* Creates string to store current input */
579                 String tempItemName;
580                 /* Creates double to store current item price */
581                 double tempItemPrice;
582                 /* For loop for user to enter all their items */
583                 for (int itemNumber = 0; itemNumber < numberOfItems; itemNumber++) {
584                         /* Prompts user each time they enter an item */
585                         System.out.print("Please enter the name of your item\n" + "=> ");
586                         /* Adds item name to tempItemName */
587                         tempItemName = scanString.nextLine();
588                         /* Prompts user to enter price of that item */
589                         System.out.print("Please enter the price of your item\n" + "=> ");
590                         /* Adds item price to tempItemPrice */
591                         tempItemPrice = scanNum.nextDouble();
592                         /* Adds item and price to itemListFormatted */
593                         itemListFormatted += String.format("%s: $%.2f\n", tempItemName, tempItemPrice);
594                         /* Adds item price to moneyTotal */
595                         moneyTotal += (int) (tempItemPrice * 100);
596                 }
597
598                 /* Computates tax in pennies */
599                 double taxrate = 0.06;
600                 int tax = (int) (moneyTotal * taxrate);
601                 int moneyTotalWithTax = moneyTotal + tax;
602
603                 /* Adds subtotal, tax and total to itemListFormatted */
604                 itemListFormatted += String.format("Subtotal: $%.2f\n" + "Tax: $%.2f\n" + "Total:
$%.2f\n\n",
605                                 ((double) (moneyTotal)) / 100, ((double) (tax)) / 100, ((double)
(moneyTotalWithTax)) / 100);
606
607                 /* Prints out purchased items with prices */
608                 System.out.print(itemListFormatted);
609
610                 /*
611                  * Asks user to submit payment and collects that payment in pennies into
612                  * variable "payment"
613                  */
614                 int payment = 0;
615                 System.out.println("Please submit your payment");
616
617                 /* Loops until user submits payment equal to or above moneyTotalWithTax */
618                 do {
619                         System.out.print("=> ");
620                         payment = (int) (scanNum.nextDouble() * 100);
621                         if (payment < moneyTotalWithTax)
622                                 System.out.printf("You still owe $%.2f%n", ((double)  (moneyTotalWithTax
- payment)) / 100);
623                 } while (payment < moneyTotalWithTax);
624
625                 /* Determine how much change of each type is required */
626                 int totalChange = payment - moneyTotalWithTax;
627                 int changeLeft = totalChange;
628                 int dollars = changeLeft / 100;
629                 changeLeft %= 100;
630                 int quarters = changeLeft / 25;
```

```
631                 changeLeft %= 25;
632                 int dimes = changeLeft / 10;
633                 changeLeft %= 10;
634                 int nickels = changeLeft / 5;
635                 changeLeft %= 5;
636                 int pennies = changeLeft;
637 
638                 /*
639                  * Outputs amount of change owed and sections it into dollars, quarters, dimes,
640                  * nickels, and pennies
641                  */
642                 System.out.printf("Thank you for your payment!\n"
643                                 + "Your change owed is $%.2f.\n"
644                                 + "Your change is %d dollars, %d quarters, %d dimes, %d nickels, and %d
pennies.",
645                                 ((double) (totalChange)) / 100, dollars, quarters, dimes, nickels,
pennies);
646         }
647 }
648 
649 /*
650  * Mitch Feigenbaum
651  * Period 5
652  * November 30, 2021
653  * On my honor, I pledge that I have neither given nor received unauthorized assistance on this
assignment or test.
654  */
655 import java.util.Scanner;
656 
657 /**
658  * This class contains methods to encrypt and decrypt strings as well as a main
659  * method which enables a user to encrypt and decrypt strings through a cli.
660  */
661 public class FeigenbaumU3 {
662         /* Defines alphabet */
663         private static String alphabetLower = "abcdefghijklmnopqrstuvwxyz";
664         private static String alphabetUpper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
665 
666         /**
667          * Creates a cli interface for encrypting or decrypting strings based on user
668          * defined parameters. This method first opens scanners to collect user input.
669          * The program then offers documentation of the program to the user with a
670          * description briefly describing program input and output. The method then
671          * proceeds to collect parameters from the user.
672          * <p>
673          * The program proceeds the ask the user to enter a message and shift. The shift
674          * is immediately moded by 26 and will prompt the user to re-enter their shift
675          * if it is 0 or a multiple of 26. Then the user will be prompted to decide
676          * whether they wish to encrypt or decrypt the string. The program will then
677          * enter a do-while loop in which the user may decide whether to encrypt or
678          * decrypt their string and will be returned their tranformed string. If the
679          * user did not enter a valid option then the loop will prompt the user again to
680          * enter a valid option.
681          *
682          * @param args added for semantics
683          * @see encrypt
684          * @see decrypt
685          */
686         public static void main(String[] args) {
687                 Scanner scanChar = new Scanner(System.in);
688                 Scanner scanNum = new Scanner(System.in);
689                 System.out.println("Program input: A string, a shift, encrypt/decrypt option");
690                 System.out.println("Program output: Either a decrypted or encrypted string based on user
input");
691                 System.out.print("Enter a string: ");
692                 String message = scanChar.nextLine();
693                 System.out.print("Enter a shift: ");
694                 int key;
695                 do {
696                         key = scanNum.nextInt() % 26;
697                         if (key == 0)
698                                 System.out.print("Please enter a number which is not a multiple of 26:
");
699                 } while (key == 0);
700                 String mode = "";
701                 do {
702                         System.out.print("Type 'e' to encrypt the string or 'd' to decrypt the string:
");
703                         mode = scanChar.nextLine().toLowerCase();
704                         if (mode.equals("e"))
705                                 System.out.printf("Your encrypted string is: %s%n", encrypt(message,
```

```
key));
                        else if (mode.equals("d"))
                                System.out.printf("Your decrypted string is: %s%n", decrypt(message,
key));
                        else
                                System.out.print("Please enter your choice again: ");
                } while (!mode.equals("e") && !mode.equals("d"));
        }

        /**
         * Returns an encrypted string based on a shift and unencrypted message. The
         * method encrypts the string by looping through the string, finding the index
         * at which the current character of the string is at in the alphabet and then
         * adds a shift to this index to find the letter which corresponds to the proper
         * shift. This letter is then added to a string which is returned to the user
         * once the string has been iterated through.
         * <p>
         * This method also contains support for lowercase and uppercase letters, which
         * allows for the passage of case-sensitive strings. In the event that a
         * non-alphabetical character is iterated upon the character is simply added to
         * the encrypted string. This method can also be used to decrypt strings by
         * passing 26 minus the shift to the key parameter. This is what the decrypt
         * method does behind the scenes.
         *
         * @param message an unencrypted string
         * @param key     a shift by which to encrypt the string
         * @return a string encrypted by a shift of key
         * @see decrypt
         */
        public static String encrypt(String message, int key) {
                String encryptedString = "";
                char currentChar;
                for (int i = 0; i < message.length(); i++) {
                        currentChar = message.charAt(i);
                        if (alphabetLower.indexOf(currentChar) != -1)
                                encryptedString += alphabetLower
                                                .charAt((alphabetLower.indexOf(currentChar) + key) % 26);
                        else if (alphabetUpper.indexOf(currentChar) != -1)
                                encryptedString += alphabetUpper
                                                .charAt((alphabetUpper.indexOf(currentChar) + key) % 26);
                        else
                                encryptedString += currentChar;
                }
                return encryptedString;
        }

        /**
         * Returns a decrypted string based on a shift and encrypted message. The method
         * uses the encrypt method to decrypt the string by using a shift of 26 minus
         * the shift used to encrypt the string.
         *
         * @param message an encrypted string
         * @param key     a shift to decrypt the string with
         * @return a string decrypted by a shift of key
         * @see encrypt
         */
        public static String decrypt(String message, int key) {
                return encrypt(message, 26 - key);
        }
}

/*
 * Mitch Feigenbaum
 * Period 5
 * February 1, 2022
 * On my honor, I pledge that I have neither given nor received unauthorized assistance on this
assignment or test.
 */
import java.util.Calendar;
import java.util.Scanner;

/**
 * The FeigenbaumU4 class is a class that provides methods for checking if a
 * given month and date are valid, getting a calendar date in ordinal form,
 * getting an astrological sign with a message, and a CLI in which a client
 * can run the program (the main method).
 */
public class FeigenbaumU4 {
        /**
         * The main method is a CLI that allows a client to interactively
         * input their birthdate to get an output of their astrological sign
```

```
784          * with a unique message. The method first creates a Calendar object.
785          * This calendar object is used to store the current month and
786          * date into the todaymonth and todayday variables. A scanner is
787          * opened to collect the clients input. Then the program enters
788          * a do while loop that prompts the client to enter their birthdate,
789          * checks that the date is valid with the checkDate method, and stores
790          * The client's birthmonth and birthdate into separate integer
791          * variables. After the client's birthmonth and birthday are defined,
792          * the program checks if it is the client's birthday and if so
793          * initializes a birthday message. Finally a print format statement is
794          * created that displays the client's age in ordinal form (using the
795          * birthdate method), a birthday message in the event that the current
796          * date is their birthday, their astrological sign, and a unique
797          * message (using the sign method)
798          * <p>
799          *
800          * @param args added for semantics
801          * @see checkDate
802          * @see birthdate
803          * @see sign
804          */
805         public static void main(String[] args) {
806                 Calendar today = Calendar.getInstance();
807                 int todaymonth = today.get(Calendar.MONTH) + 1;
808                 int todayday = today.get(Calendar.DAY_OF_MONTH);
809                 Scanner scanNum = new Scanner(System.in);
810                 int birthMonth;
811                 int birthDay;
812                 do {
813                         System.out.print("What month were you born in? (number): ");
814                         birthMonth = scanNum.nextInt();
815                         System.out.print("What day (number): ");
816                         birthDay = scanNum.nextInt();
817                         if (!checkDate(birthMonth, birthDay))
818                                 System.err.println("Error: date does not exist.");
819                 } while (!checkDate(birthMonth, birthDay));
820                 String birthDayMessage = "";
821                 if (birthMonth == todaymonth && birthDay == todayday)
822                         birthDayMessage = "Happy Birthday to you!";
823                 System.out.printf("Your birthday is:\t%s\t%s%n%s",
824                                 birthdate(birthMonth, birthDay), birthDayMessage, sign(birthMonth,
birthDay));
825         }
826
827         /**
828          * The birthdate method is used to return a textual version of a date
829          * based on an integer representation of the month and date. First an
830          * array of all 12 calendar months is initialized. then a String
831          * format statement is returned that contains a textual representation
832          * of the month, and an ordinal representation of the date. (using the
833          * toOrdinal method)
834          * <p>
835          *
836          * @param m An integer representation of the month
837          * @param d An integer representation of the date
838          * @return a formatted string containing the month and date
839          * @see toOrdinal
840          */
841         public static String birthdate(int m, int d) {
842                 String[] months = {
843                                 "January",
844                                 "February",
845                                 "March",
846                                 "April",
847                                 "May",
848                                 "June",
849                                 "July",
850                                 "August",
851                                 "September",
852                                 "October",
853                                 "November",
854                                 "December",
855                 };
856                 return String.format("%s %s", months[m - 1], toOrdinal(d));
857         }
858
859         /**
860          * The method sign provides an astrological sign and horoscope based
861          * on a provided month and date. The method iterates through a list of
862          * if statements that return the astrological sign and horoscope of a
863          * certain character (eg Aquarius) provided that the month and date
```

```
 * are within a certain range of dates in which the character is
 * valid. If the criteria of an if statement is fulfilled the method
 * returns a string with an astrological sign and horoscope formatted
 * with tabs for alignment. The tabstops are meant to be used in
 * conjunction with the main method's print format statement which
 * also displays the users birthday and a happy birthday message.
 * <p>
 *
 * @param m An integer representation of the month
 * @param d An integer representation of the date
 * @return A string with the astrological sign and horoscope for the
 *         provided month and day
 * @see main
 */
public static String sign(int m, int d) {
        return (m == 1 && d >= 20) || (m == 2 && d <= 18)
                ? "Your sign is:\t\tAquarius\n" + "Horoscope:\t\tYou must hold back your desires and temptations."
                : (m == 2 && d >= 19) || (m == 3 && d <= 20)
                ? "Your sign is:\t\tPisces\n" + "Horoscope:\t\tSoon you will make amends with all you have wronged."
                : (m == 3 && d >= 21) || (m == 4 && d <= 19)
                ? "Your sign is:\t\tAries\n" + "Horoscope:\t\tYour home planet will be destroyed."
                : (m == 4 && d >= 20) || (m == 5 && d <= 20)
                ? "Your sign is:\t\tTaurus\n" + "Horoscope:\t\tHave faith in the plan."
                : (m == 5 && d >= 21) || (m == 6 && d <= 20)
                ? "Your sign is:\t\tGemini\n" + "Horoscope:\t\tThe best days of your life are already over."
                : (m == 6 && d >= 21) || (m == 7 && d <= 22)
                ? "Your sign is:\t\tCancer\n" + "Horoscope:\t\tYou will hold back your own success."
                : (m == 7 && d >= 23) || (m == 8 && d <= 22)
                ? "Your sign is:\t\tLeo\n" + "Horoscope:\t\tYou will achieve great success in the textile industry."
                : (m == 8 && d >= 23) || (m == 9 && d <= 22)
                ? "Your sign is:\t\tVirgo\n" + "Horoscope:\t\tThere is light at the end of your sorrow."
                : (m == 9 && d >= 23) || (m == 10 && d <= 22)
                ? "Your sign is:\t\tLibra\n" + "Horoscope:\t\tTerrible things are going to happen."
                : (m == 10 && d >= 23) || (m == 11 && d <= 21)
                ? "Your sign is:\t\tScorpio\n" + "Horoscope:\t\tYour days are numbered. Keep watch of all enemies."
                : (m == 11 && d >= 22) || (m == 12 && d <= 21)
                ? "Your sign is:\t\tSagittarius\n" + "Horoscope:\t\tSomeday you will travel into the multiverse."
                : (m == 12 && d >= 22) || (m == 1 && d <= 19)
                ? "Your sign is:\t\tCapricorn\n" + "Horoscope:\t\tYou will achieve inner peace."
                : null;
        }

        /**
         * The toOrdinal method can convert a number (from 1 to 39) from
         * integer form into ordinal form. Based on a number's position in a
         * certain range (eg 11-19), an element is chosen from an array which
         * represents a number's ordinal form. If a number is not composed of
         * only one word (eg twenty-sixth), a format string is returned
         * containing the numbers tenth place, a dash, and the singular
         * number that represents the digit in its ones place.
         * <p>
         *
         * @param d a digit 1-39
         * @return an ordinal number
         */
        public static String toOrdinal(int d) {
                String[] singleDays = {
                                "first",
                                "second",
                                "third",
                                "fourth",
                                "fifth",
                                "sixth",
                                "seventh",
                                "eighth",
                                "ninth",
                };
                String[] teens = {
                                "eleventh",
                                "twelfth",
                                "thirteenth",
```

```
                                "fourteenth",
                                "fifteenth",
                                "sixteenth",
                                "seventeenth",
                                "eighteenth",
                                "nineteenth",
                };
                String[] tenths = {
                                "tenth",
                                "twentieth",
                                "thirtieth",
                };
                String[] tens = {
                                "twenty",
                                "thirty",
                };
                return (d < 10)
                                ? singleDays[d - 1]
                                : (d > 10 && d < 20)
                                ? teens[d - 11]
                                : (d % 10 == 0)
                                ? tenths[d / 10 - 1]
                                : String.format("%s-%s", tens[d / 10 - 2], singleDays[d % 10 - 1]);
        }

        /**
         * The checkDate method verifies whether a current month and date are
         * possible calendar positions. It iterates through a series of
         * impossible criteria that will return false if they are applicable
         * to the provided month and day. The date is first checked that
         * the month is between 1 and 12 and that the date is at least one.
         * Then the date is checked to verify that it is not too large for the
         * current calendar month. If none of the unverifiable criteria are
         * met, the method will return true
         * <p>
         *
         * @param m the current month in integer form
         * @param d the current day in integer form
         * @return true if the provided month and day are valid
         */
        public static boolean checkDate(int m, int d) {
                return !(m < 1
                                || m > 12
                                || d < 1
                                || d > 31
                                || ((m == 4
                                                || m == 6
                                                || m == 9
                                                || m == 11)
                                                && d > 30)
                                || (m == 2
                                                && d > 29));
        }
}
/*
*Name: Mitch Feigenbaum
*Date: 8/27/2021
*Period: 5
*
*Program Description:
*
*Algorithm:
*
*On my honor, by submitting this code I am claiming that it was written
and tested by me.
*Any help I received was in the form of asking questions for clarification
and no direct copying was done.
*
*/
import java.io.*;
public class HelloWorld
{
   public static void main(String[] args)
   {
      System.out.println("Hello, world!"); // Prints hello world to standard output
   }
}
```