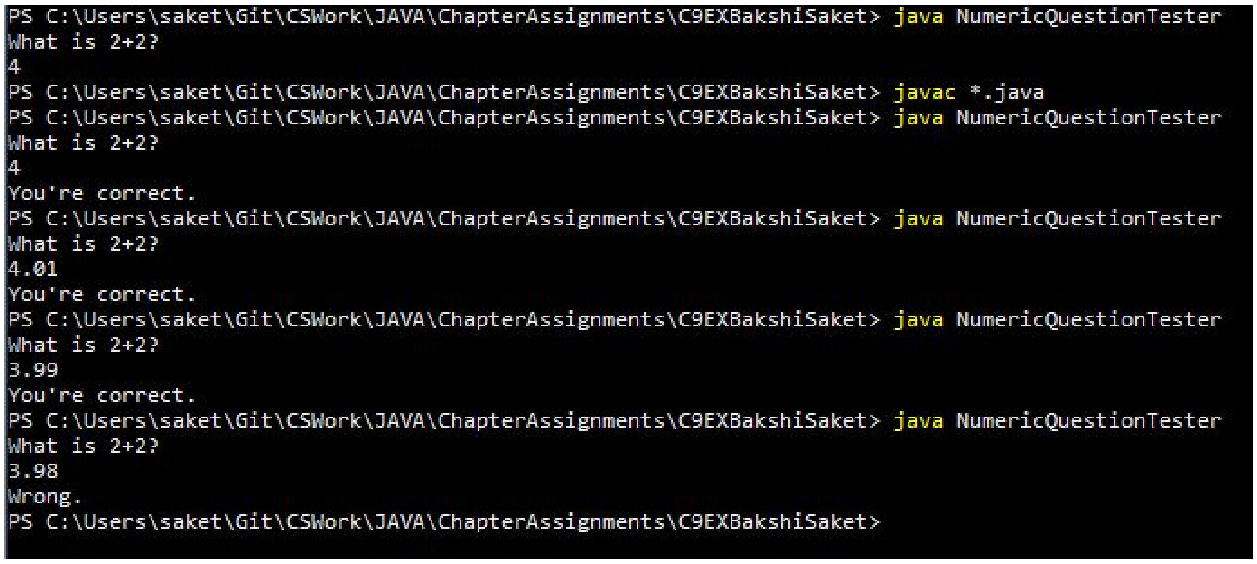
Exercise 9.1

```
/** Saket Bakshi. 2/10/19. Period 6. This is used for question 1 of Chapter 9.
	A question with a text and an answer.
*/
public class Question
{
	private String text;
	private String answer;

	/**
		Constructs a question with empty question and answer.
	*/
	public Question()
	{
		text = "";
		answer = "";
	}

	/**
		Sets the question text
		@param questionText the text of this question
	*/
	public void setText(String questionText)
	{
		text = questionText;
	}

	/**
		Sets the answer for this question.
		@param correctResponse the answer
	*/
	public void setAnswer(String correctResponse)
	{
		answer = correctResponse;
	}

	/**
		Checks a given response for correctness.
		@param response the response to check
		@return true if the response was correct, false otherwise
	*/
	public boolean checkAnswer(String response)
	{
		return response.equals(answer);
	}

	/**
		Displays this question
	*/
	public void display()
	{
		System.out.println(text);
	}
}
/** Saket Bakshi. 2/10/19. Period 6. This is used for question 1 of Chapter 9.
	A numeric question with a text and an answer where approximations are ok.
*/
public class NumericQuestion extends Question
{
	private double answer;

	/**
		Constructs a question with empty question and answer.
	*/
	public NumericQuestion()
	{
		super();
	}

	/**
		Sets the answer for this question.
		@param correctResponse the answer
	*/
	public void setAnswer(double correctResponse)
	{
		answer = correctResponse;
	}

	/**
	Checks a given response for correctness.
	@param response the response to check
	@return true if the response was correct, false otherwise
	*/
	public boolean checkAnswer(double response)
	{
		if(Math.abs(response - answer) <= 0.01)
```

```java
				return true;
			else
				return false;
		}
}
/** Saket Bakshi. 2/10/19. Period 6. This is used for question 1 of Chapter 9.
		Tests the NumericQuestion class.
*/
import java.util.Scanner;

public class NumericQuestionTester
{
		public static void main(String[] args)
		{
			NumericQuestion tester = new NumericQuestion();
			tester.setText("What is 2+2?");
			tester.setAnswer(4);
			tester.display();
			Scanner key = new Scanner(System.in);
			double answered = key.nextDouble();
			if(tester.checkAnswer(answered))
				System.out.println("You're correct.");
			else
				System.out.println("Wrong.");
		}
}
```

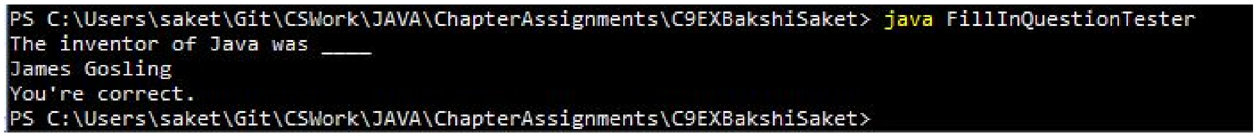


```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket> java NumericQuestionTester
What is 2+2?
4
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket> javac *.java
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket> java NumericQuestionTester
What is 2+2?
4
You're correct.
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket> java NumericQuestionTester
What is 2+2?
4.01
You're correct.
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket> java NumericQuestionTester
What is 2+2?
3.99
You're correct.
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket> java NumericQuestionTester
What is 2+2?
3.98
Wrong.
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket>
```

Exercise 9.2

```
/** Saket Bakshi. 2/10/19. Period 6. This is used for question 2 of Chapter 9.
	A fill-in-the-blank question with a text and an answer.
*/
public class FillInQuestion extends Question
{
	/**
		Constructs a question with empty question and answer.
	*/
	public FillInQuestion()
	{
		super();
	}

	/**
		Constructs a question with empty question and answer.
		@param questionAndAnswer the fill-in-the-blank with the answer filled out
	*/
	public FillInQuestion(String questionAndAnswer)
	{
		int beginningOfAnswer = questionAndAnswer.indexOf("_");
		super.setText(questionAndAnswer.substring(0, beginningOfAnswer) + "____");

		int endOfAnswer = questionAndAnswer.indexOf("_", beginningOfAnswer + 1);
		super.setAnswer(questionAndAnswer.substring(beginningOfAnswer + 1,
endOfAnswer));
	}
}
/** Saket Bakshi. 2/10/19. Period 6. This is used for question 2 of Chapter 9.
	Tests the FillInQuestion class.
*/
import java.util.Scanner;

public class FillInQuestionTester
{
	public static void main(String[] args)
	{
		FillInQuestion tester = new FillInQuestion("The inventor of Java was _James
Gosling_");
		tester.display();

		Scanner key = new Scanner(System.in);
		String answer = key.next();
```

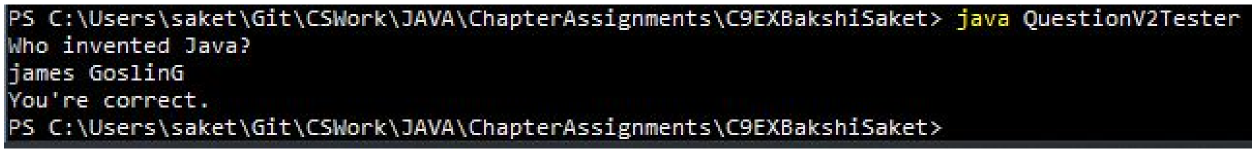
```
			if(tester.checkAnswer(answer))
				System.out.println("You're correct.");
			else
				System.out.println("Wrong.");
	}
}
```



```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket> java FillInQuestionTester
The inventor of Java was ____
James Gosling
You're correct.
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket>
```

Exercise 9.3

```
/** Saket Bakshi. 2/10/19. Period 6. This is used for question 3 of Chapter 9.
	A question with a text and an answer. Answer is lenient with lower and upper case.
*/
public class QuestionV2
{
	private String text;
	private String answer;

	/**
		Constructs a question with empty question and answer.
	*/
	public QuestionV2()
	{
		text = "";
		answer = "";
	}

	/**
		Sets the question text
		@param questionText the text of this question
	*/
	public void setText(String questionText)
	{
		text = questionText;
	}

	/**
		Sets the answer for this question. Is lenient with upper and lowercase.
		@param correctResponse the answer
	*/
	public void setAnswer(String correctResponse)
	{
		answer = correctResponse.toLowerCase();
	}

	/**
		Checks a given response for correctness.  Is lenient with upper and lowercase.
		@param response the response to check
		@return true if the response was correct, false otherwise
	*/
	public boolean checkAnswer(String response)
	{
		String reply = response.toLowerCase();
		return reply.equals(answer.toLowerCase());
	}

	/**
		Displays this question
	*/
	public void display()
	{
		System.out.println(text);
	}

	public String getAnswer() {return answer;}
}
/** Saket Bakshi. 2/10/19. Period 6. This is used for question 3 of Chapter 9.
	Tests the QuestionV2 class.
*/
import java.util.Scanner;

public class QuestionV2Tester
{
	public static void main(String[] args)
	{
		QuestionV2 tester = new QuestionV2();
		tester.setText("Who invented Java?");
		tester.setAnswer("James Gosling");
		tester.display();
		Scanner key = new Scanner(System.in);
		String answered = key.next();
		if(tester.checkAnswer(answered))
			System.out.println("You're correct.");
		else
			System.out.println("Wrong.");
	}
}
```



```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket> java QuestionV2Tester
Who invented Java?
james GoslinG
You're correct.
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket>
```

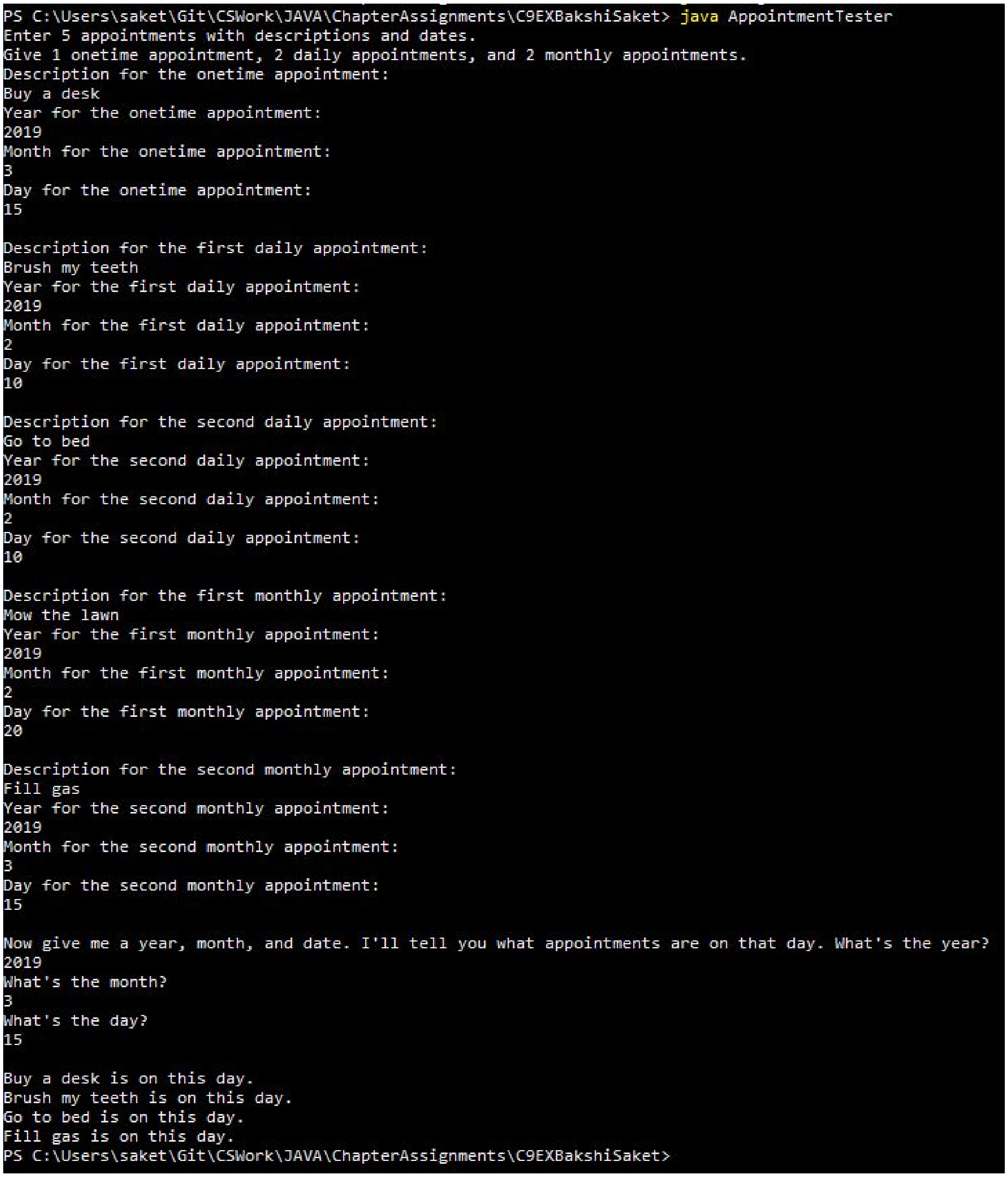
Project 9.1

```
/** Saket Bakshi. 2/10/19. Period 6. This is used for project 1 of Chapter 9.
	Makes appointments with dates and descriptions.
*/
public class Appointment
{
	private String description;

	private int year;
	private int month;
	private int day;

	/**
		Makes appointments with dates and descriptions.
	*/
	public Appointment()
	{
		description = "";
		year = 0;
		month = 0;
		day = 0;
	}

	/**
		Makes appointments with dates and descriptions.
		@param description the description
		@param year the year
		@param month the month
		@param day the day
	*/
	public Appointment(String description, int year, int month, int day)
	{
		this.description = description;
		this.year = year;
		this.month = month;
		this.day = day;
	}

	/**
		Checks if an appointment occurs on a given day.
		@param year the year to check for
		@param month the month to check for
		@param day the day to check for
		@return if an appointment occurs on a given day
	*/
	public boolean occursOn(int year, int month, int day)
	{
		if(this.year == year && this.month == month && this.day == day)
			return true;
		else
			return false;
	}

	/**
		Returns the year
		@return the year
	*/
	public int getYear() {return year;}

	/**
		Returns the month
		@return the month
	*/
	public int getMonth() {return month;}

	/**
		Returns the day
		@return the day
	*/
	public int getDay() {return day;}

	/**
		Returns the description
		@return the description
	*/
	public String getDescription() {return description;}

	/**
		Sets the year
		@param y the year
	*/
	public void setYear(int y) {year = y;}

	/**
		Sets the month
		@param m the month
	*/
	public void setMonth(int m) {month = m;}

	/**
		Sets the day
		@param d the day
	*/
	public void setDay(int d) {day = d;}

	/**
		Sets the description
		@param d the description
	*/
	public void setDescription(String d) {description = d;}
}
/** Saket Bakshi. 2/10/19. Period 6. This is used for project 1 of Chapter 9.
	Makes onetime appointments with dates and descriptions.
*/
public class Onetime extends Appointment
{
	/**
		Makes onetime appointments with dates and descriptions.
	*/
	public Onetime()
	{
		super();
	}

	/**
		Makes appointments with dates and descriptions.
		@param description the description
		@param year the year
		@param month the month
		@param day the day
	*/
	public Onetime(String description, int year, int month, int day)
	{
		super(description, year, month, day);
	}
}
/** Saket Bakshi. 2/10/19. Period 6. This is used for project 1 of Chapter 9.
	Makes daily appointments with dates and descriptions.
*/
```

```
public class Daily extends Appointment
{
	/**
		Makes onetime appointments with dates and descriptions.
	*/
	public Daily()
	{
		super();
	}

	/**
		Makes appointments with dates and descriptions.
		@param description the description
		@param year the year
		@param month the month
		@param day the day
	*/
	public Daily(String description, int year, int month, int day)
	{
		super(description, year, month, day);
	}

	/**
		Checks if an appointment occurs on a given day.
		@param year the year to check for
		@param month the month to check for
		@param day the day to check for
		@return if an appointment occurs on a given day
	*/
	public boolean occursOn(int year, int month, int day)
	{
		return true;
	}
}
/** Saket Bakshi. 2/10/19. Period 6. This is used for project 1 of Chapter 9.
	Makes monthly appointments with dates and descriptions.
*/
public class Monthly extends Appointment
{
	/**
		Makes onetime appointments with dates and descriptions.
	*/
	public Monthly()
	{
		super();
	}

	/**
		Makes appointments with dates and descriptions.
		@param description the description
		@param year the year
		@param month the month
		@param day the day
	*/
	public Monthly(String description, int year, int month, int day)
	{
		super(description, year, month, day);
	}

	/**
		Checks if an appointment occurs on a given day.
		@param year the year to check for
		@param month the month to check for
		@param day the day to check for
		@return if an appointment occurs on a given day
	*/
	public boolean occursOn(int year, int month, int day)
	{
		if(super.getDay() == day)
			return true;
		else
			return false;
	}
}
/** Saket Bakshi. 2/10/19. Period 6. This is used for project 1 of Chapter 9.
	Tests the Appointment class and its subclasses.
*/
import java.util.Scanner;
public class AppointmentTester
{
	public static void main(String[] args)
	{
		System.out.println("Enter 5 appointments with descriptions and dates.\nGive 1
onetime appointment, 2 daily appointments, and 2 monthly appointments.");
		Scanner key = new Scanner(System.in);
```

```
System.out.println("Description for the onetime appointment: ");
String des1 = key.nextLine();
System.out.println("Year for the onetime appointment: ");
int year1 = key.nextInt();
System.out.println("Month for the onetime appointment: ");
int month1 = key.nextInt();
System.out.println("Day for the onetime appointment: ");
int day1 = key.nextInt();
Onetime first1 = new Onetime(des1, year1, month1, day1);
key.nextLine();
System.out.println();

System.out.println("Description for the first daily appointment: ");
String des2 = key.nextLine();
System.out.println("Year for the first daily appointment: ");
int year2 = key.nextInt();
System.out.println("Month for the first daily appointment: ");
int month2 = key.nextInt();
System.out.println("Day for the first daily appointment: ");
int day2 = key.nextInt();
Daily first2 = new Daily(des2, year2, month2, day2);
key.nextLine();
System.out.println();

System.out.println("Description for the second daily appointment: ");
String des3 = key.nextLine();
System.out.println("Year for the second daily appointment: ");
int year3 = key.nextInt();
System.out.println("Month for the second daily appointment: ");
int month3 = key.nextInt();
System.out.println("Day for the second daily appointment: ");
int day3 = key.nextInt();
Daily second3 = new Daily(des3, year3, month3, day3);
key.nextLine();
System.out.println();

System.out.println("Description for the first monthly appointment: ");
String des4 = key.nextLine();
System.out.println("Year for the first monthly appointment: ");
int year4 = key.nextInt();
System.out.println("Month for the first monthly appointment: ");
int month4 = key.nextInt();
System.out.println("Day for the first monthly appointment: ");
```

```
		int day4 = key.nextInt();
		Monthly first4 = new Monthly(des4, year4, month4, day4);
		key.nextLine();
		System.out.println();

		System.out.println("Description for the second monthly appointment: ");
		String des5 = key.nextLine();
		System.out.println("Year for the second monthly appointment: ");
		int year5 = key.nextInt();
		System.out.println("Month for the second monthly appointment: ");
		int month5 = key.nextInt();
		System.out.println("Day for the second monthly appointment: ");
		int day5 = key.nextInt();
		Monthly second5 = new Monthly(des5, year5, month5, day5);
		key.nextLine();
		System.out.println();

		System.out.println("Now give me a year, month, and date. I'll tell you what
appointments are on that day. What's the year? ");
		int yearTest = key.nextInt();
		System.out.println("What's the month? ");
		int monthTest = key.nextInt();
		System.out.println("What's the day? ");
		int dayTest = key.nextInt();
		System.out.println();

		if(first1.occursOn(yearTest, monthTest, dayTest))
			System.out.println(first1.getDescription() + " is on this day.");

		if(first2.occursOn(yearTest, monthTest, dayTest))
			System.out.println(first2.getDescription() + " is on this day.");
		if(second3.occursOn(yearTest, monthTest, dayTest))
			System.out.println(second3.getDescription() + " is on this day.");

		if(first4.occursOn(yearTest, monthTest, dayTest))
			System.out.println(first4.getDescription() + " is on this day.");
		if(second5.occursOn(yearTest, monthTest, dayTest))
			System.out.println(second5.getDescription() + " is on this day.");
	}
}
```



```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket> java AppointmentTester
Enter 5 appointments with descriptions and dates.
Give 1 onetime appointment, 2 daily appointments, and 2 monthly appointments.
Description for the onetime appointment:
Buy a desk
Year for the onetime appointment:
2019
Month for the onetime appointment:
3
Day for the onetime appointment:
15

Description for the first daily appointment:
Brush my teeth
Year for the first daily appointment:
2019
Month for the first daily appointment:
2
Day for the first daily appointment:
10

Description for the second daily appointment:
Go to bed
Year for the second daily appointment:
2019
Month for the second daily appointment:
2
Day for the second daily appointment:
10

Description for the first monthly appointment:
Mow the lawn
Year for the first monthly appointment:
2019
Month for the first monthly appointment:
2
Day for the first monthly appointment:
20

Description for the second monthly appointment:
Fill gas
Year for the second monthly appointment:
2019
Month for the second monthly appointment:
3
Day for the second monthly appointment:
15

Now give me a year, month, and date. I'll tell you what appointments are on that day. What's the year?
2019
What's the month?
3
What's the day?
15

Buy a desk is on this day.
Brush my teeth is on this day.
Go to bed is on this day.
Fill gas is on this day.
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C9EXBakshiSaket>
```