

4.1 Ethical and Social Issues Around Data Collection

Self-Study Guide · About 45 minutes · With answer key

By the end you will be able to

- Explain why collecting and storing personal data puts people's privacy at risk.
- Describe steps a programmer can take to protect the privacy of a program's users.
- Define algorithmic bias and tell it apart from an ordinary one-time bug.
- Explain how the way data was collected can make conclusions drawn from it unfair or wrong.
- Explain how incomplete or inaccurate data makes a program work incorrectly or inefficiently.
- Decide whether a data set is a good fit for the question you want to answer.

What to remember from earlier

- Every program has impacts on society, the economy, and culture. ([Topic 3.2: Impact of Program Design, 3.2.1 Programs Affect People](#))
- A program can cause harm its designers never intended. ([Topic 3.2: Impact of Program Design, 3.2.2 Unintended Harmful Effects](#))
- A reliable program works under all its stated conditions. ([Topic 3.2: Impact of Program Design, 3.2.3 System Reliability](#))
- Testers choose typical, boundary, and special cases. ([Topic 3.2: Impact of Program Design, 3.2.4 Testing With a Variety of Conditions](#))

4.1.1 Personal Data and Privacy Risks

This unit is about programs that work with many values at once: hundreds of test scores, a year of temperatures, every student in a school. Before learning how to store and process that much data, it is worth asking where the data comes from. Very often it comes from people, and that means a program that collects it is holding something that belongs, in a real sense, to someone else.

A **data set** is a collection of related pieces of information, such as the sign-up forms for a club or the readings from a weather sensor. When the information is about specific people, it is **personal data**: anything that tells you about a particular person, such as their name, home address, phone number, birthday, location, grades, or health. **Privacy** is a person's ability to control who knows that information about them.

Whenever personal data is typed into a computer, privacy is at risk. Once a program collects and stores the data, it can be:

- **seen by people who should not see it**, for example if the stored file is copied, a device is lost, or someone breaks into the system;
- **used for a purpose the person never agreed to**, such as selling a list of email addresses collected "only for club reminders";

- **combined with other data** to reveal more than any single piece does;
- **kept long after it is needed**, so the risk lasts for years.

None of these risks require a bug in the code. A program can work perfectly and still put people's privacy at risk, simply because it holds their data.

KEY IDEA

Collecting and storing personal data on a computer puts people's privacy at risk, even when the program works correctly. The more personal data a program holds, and the longer it holds it, the more there is to lose.

WORKED EXAMPLE What a club sign-up app collects

A hypothetical app lets students sign up for after-school clubs. Its form asks for six things. For each one, ask two questions: does the app need it to do its job, and what could happen if it were exposed?

Field	Needed to sign up for a club?	Risk if exposed
Name	Yes: the club leader needs a roster	Low on its own
School email	Yes: for meeting reminders	Unwanted messages or scams sent to the student
Grade level	Yes: some clubs are for certain grades	Low on its own
Home address	No	Anyone with the data knows where the student lives
Birthday	No	Can help someone pretend to be the student
Live phone location	No	Anyone with the data can follow where the student goes

Only the first three fields serve the app's purpose. The last three add serious risk and give nothing back to the student.

Now vary one thing: look at pieces that seem harmless alone. A birthday by itself matches thousands of people. A grade level matches a few hundred students. But "born on March 3, in 10th grade, in the robotics club" may match exactly one student. Pieces of data that reveal little separately can **identify a specific person** when they are combined. That is why "each field is harmless" is not the same as "the data set is harmless."

WATCH OUT

"I have nothing to hide, so privacy does not matter." Students often think privacy is only about secrets. But the risks above are about safety and control: an address that lets a stranger find you, an email list sold without asking. Every user is affected by these risks, not only people with something to hide.

PAUSE AND TRY**Question 1** Multiple choice

A hypothetical homework-planner app stores each student's name, class schedule, and the times they are usually at home alone. Which statement best describes the privacy risk?

- A. There is no risk, because the app has no bugs.
- B. There is no risk, because a class schedule is not secret.
- C. If the stored data were exposed, someone could learn where and when a specific student is alone.
- D. The only risk is that the app might run slowly with so much data.

Question 2 Short answer

A hypothetical running app posts each user's route map publicly, but never shows the user's address. Explain how a public route map could still reveal where someone lives.

4.1.2 Safeguarding Users' Privacy

If holding personal data creates risk, the programmer who designs the program decides how much risk the users carry. Users usually cannot see the code; they have to trust that whoever wrote it was careful. So when developing a new program, a programmer should try to **safeguard** (protect) the personal privacy of the people who use it.

Good habits that do this include:

Habit	What it means
Collect only what is needed	Ask for a field only if the program's purpose requires it
Tell users and ask first	Say what is collected and why, and get the user's agreement
Use data only for the stated purpose	Data collected for reminders is used for reminders, nothing else
Show and share as little as possible	Display only the part of a value a person needs to see
Limit who can reach stored data	Only the people and programs that need the data can open it
Delete data when it is no longer needed	Data that no longer exists cannot be exposed

The first habit is the strongest: data that was never collected can never be exposed, misused, or combined.

KEY IDEA

Programmers should design programs to protect the privacy of their users: collect only the personal data the program needs, use it only for its stated purpose, show as little of it as possible, and delete it when it is no longer needed.

WORKED EXAMPLE Showing only part of a phone number

A hypothetical dentist-reminder program confirms where a text reminder will go. The first version prints the whole phone number on the screen, where anyone standing nearby can read it.

```
1 String phone = "5550198342";
2 String lastFour = phone.substring(phone.length() - 4);
3 System.out.println("Phone on file: " + phone);
4 System.out.println("Reminder goes to the number ending in " + lastFour);
```

```
Phone on file: 5550198342
Reminder goes to the number ending in 8342
```

The user only needs enough to recognize their own number, so the full number on line 3 shows more than necessary. The improved version removes that line and keeps only the last four digits:

```
1 String phone = "5550198342";
2 String lastFour = phone.substring(phone.length() - 4);
3 System.out.println("Reminder goes to the number ending in " + lastFour);
```

```
Reminder goes to the number ending in 8342
```

Trace line 2: `phone.length()` is 10, so `phone.length() - 4` is 6. `substring(6)` returns the characters from index 6 to the end, which are "8342". The patient can still check that the number is theirs, but a stranger reading the screen learns almost nothing.

Now vary the kind of data. An email address can be shortened the same way, keeping just enough for the owner to recognize it:

```
1 String email = "jordan.lee@example.com";
2 int atSign = email.indexOf("@");
3 String masked = email.substring(0, 1) + "***" + email.substring(atSign);
4 System.out.println("Code sent to " + masked);
```

```
Code sent to j***@example.com
```

`indexOf("@")` finds the position of the @ (index 10). `substring(0, 1)` is "j", and `substring(atSign)` is "@example.com". The program puts "***" between them. The idea is the same as before: display the smallest piece that does the job.

WATCH OUT

"Collect everything now, just in case it is useful later." This feels like good planning, but every extra field is extra risk for the user with no benefit today. In the club sign-up example, asking for home addresses "just in case" means a leak would expose where every member lives. Collect a field only when the program needs it now.

A related mix-up: declaring an instance variable `private` (Topic 3.8) controls which **code** can use it. It does not keep the stored data away from **people**. Protecting privacy is about what you collect, show, keep, and share.

PAUSE AND TRY**Question 3** Tracing

A hypothetical locker program prints a confirmation. What does it print?

```
1 String studentId = "40917325";
2 String shown = "****" + studentId.substring(4);
3 System.out.println("Locker assigned to student " + shown);
```

Question 4 Multiple choice

A programmer is designing a hypothetical app that tells students when the school library is busy. Which design choice best safeguards users' privacy?

- A. Record each student's name and exact arrival time, and keep the records forever in case they are useful later.
- B. Count how many people are in the library, without recording who they are.
- C. Record each student's location all day so the busy times are more accurate.
- D. Store names, but make the variable that holds them `private`.

4.1.3 Algorithmic Bias

Some harms from data and programs are not about privacy at all. They are about fairness: a program that treats one group of people worse than everyone else, over and over.

Recall from Topic 3.2 that a program can work perfectly for typical users and still fail people its designers did not picture. Now imagine that failure is not random. It always lands on the same kind of person. **Algorithmic bias** describes errors in a program that are **systemic** (built into how the program works, not caused by chance) and **repeated** (they happen every time the situation comes up), and that create **unfair outcomes for a specific group** of users.

Compare that with an ordinary glitch. If a server crashes for ten minutes, whoever happened to be online is affected; next time it might be anyone else. That is a reliability problem, but it is not bias. Bias is when you can predict who will be treated unfairly, because the cause is in the program's rules or the data behind them.

KEY IDEA

Algorithmic bias is a systemic, repeated error in a program that produces unfair results for a particular group of users. The key test is whether the same group is harmed every time.

WORKED EXAMPLE A sign-up form that rejects short names

A hypothetical sign-up program checks each last name before accepting it. The programmer assumed every real last name has at least three letters, so line 11 rejects anything shorter. Watch what happens with five sign-ups.

NameCheck.java

```

1  import java.util.Scanner;
2
3  public class NameCheck
4  {
5      public static void main(String[] args)
6      {
7          Scanner signUps = new Scanner("Rivera Ng Okafor Li Thompson");
8          for (int person = 1; person <= 5; person++)
9          {
10             String lastName = signUps.next();
11             if (lastName.length() >= 3)
12             {
13                 System.out.println(lastName + ": accepted");
14             }
15             else
16             {
17                 System.out.println(lastName + ": rejected, name too short");
18             }
19         }
20     }
21 }

```

```

Rivera: accepted
Ng: rejected, name too short
Okafor: accepted
Li: rejected, name too short
Thompson: accepted

```

lastName	lastName.length()	length() >= 3	Result
Rivera	6	true	accepted
Ng	2	false	rejected
Okafor	6	true	accepted
Li	2	false	rejected
Thompson	8	true	accepted

The program compiles and runs without crashing. It even does exactly what the programmer wrote. But two-letter last names are real and common in some families and cultures, and every person with one is turned away, every time they try. That makes the error **systemic** (it comes from the rule on line 11), **repeated** (it happens on every attempt), and **unfair to a specific group** (people with short last names). It is algorithmic bias.

The fix is to remove the wrong assumption. Changing line 11 to `if (lastName.length() >= 1)` still rejects an empty entry but accepts every real name; with that change all five names print accepted.

Now vary where the bias comes from. A hypothetical delivery app decides who gets same-day delivery using a list of neighborhoods where past orders arrived on time. Neighborhoods where the company never delivered before have no past orders, so they are never on the list. Every resident of those areas is denied same-day delivery, again and again, even though no line of code

mentions them. Here the unfair rule was not typed in by hand; it came from the data the program relied on. Bias can enter through the code's rules or through the data, and the next subsections look at the data side.

WATCH OUT

"The program just follows the rules, so it cannot be unfair." Students often think a computer is automatically neutral. But the rules were written by people, and the data came from people. A program applies an unfair assumption faster and more consistently than any person could, so it can make the unfairness worse, not better.

PAUSE AND TRY

Question 5 Multiple choice

Which situation is the best example of algorithmic bias?

- A. A game crashes once during a storm when the power flickers.
- B. A photo app's face-detection feature fails repeatedly for users with darker skin, because it was built using mostly photos of lighter-skinned people.
- C. A calculator app has a typo in its help screen.
- D. A music app recommends a song a user does not like.

Question 6 Debugging

A hypothetical school form rejects any last name that contains a space, because the programmer assumed names are one word.

SpaceCheck.java

```
1 import java.util.Scanner;
2
3 public class SpaceCheck
4 {
5     public static void main(String[] args)
6     {
7         Scanner signUps = new Scanner("Garcia\nDe La Cruz\nPark\nVan Dyke");
8         for (int person = 1; person <= 4; person++)
9         {
10             String lastName = signUps.nextLine();
11             if (lastName.indexOf(" ") == -1)
12             {
13                 System.out.println(lastName + ": accepted");
14             }
15             else
16             {
17                 System.out.println(lastName + ": rejected, invalid name");
18             }
19         }
20     }
21 }
```

```
Garcia: accepted  
De La Cruz: rejected, invalid name  
Park: accepted  
Van Dyke: rejected, invalid name
```

Explain why this is algorithmic bias, and correct line 11 so every non-empty name is accepted.

4.1.4 How the Data Was Collected

Programmers often use data to learn something new: which feature users want, what the average score was, how busy a place will be next week. Using data to predict values or reach conclusions beyond what was directly measured is called **extrapolating**. Every conclusion like that is only as good as the way the data was gathered.

The **collection method** is how a data set was gathered: who was asked or measured, where, when, and how. If the method reaches some kinds of people or situations more than others, the data set does not represent the whole group, and any conclusion about the whole group can be wrong. That tilt in the data is called **bias**, the same word as in algorithmic bias, because a program built on tilted data tends to produce tilted results. So before using data to draw conclusions, a programmer should ask how it was collected and whether that method could be biased.

KEY IDEA

Before using a data set to draw conclusions or extrapolate new information, find out how the data was collected and ask whether that method could be biased toward some people or situations.

WORKED EXAMPLE A survey taken in the gym

A hypothetical student council wants to know whether the whole school wants the gym to stay open later. A council member surveys 40 students one afternoon, standing in the gym. The program below turns the results into a percent.

```
1 int surveyed = 40;  
2 int wantLaterHours = 36;  
3 int percent = wantLaterHours * 100 / surveyed;  
4 System.out.println(percent + "% want later gym hours");
```

```
90% want later gym hours
```

Trace line 3: $36 * 100$ is 3600, and $3600 / 40$ is 90. The arithmetic is correct. The problem is the collection method. Students who are in the gym in the afternoon are the students who already use the gym, so of course most of them want it open later. The survey never reached students who go home, work, or are at other activities. Concluding "90% of the school wants later hours" extrapolates from a biased group to everyone.

Now vary only the collection method. Suppose instead the council hands the same question to every homeroom, reaching 400 students from every part of the school:

```
1 int surveyed = 400;
2 int wantLaterHours = 120;
3 int percent = wantLaterHours * 100 / surveyed;
4 System.out.println(percent + "% want later gym hours");
```

```
30% want later gym hours
```

Same question, same program, same arithmetic ($120 * 100 / 400$ is 30), and a very different answer. Nothing about the code changed. The only difference is who the data came from. A correct program cannot fix data that was collected in a biased way.

WATCH OUT

"More responses means the data is accurate." Students often trust a big number of responses. But if a survey is only posted where one kind of person sees it, collecting 4,000 responses just repeats the same tilt 4,000 times. What matters first is who the method can reach, not only how many answers came in.

PAUSE AND TRY

Question 7 Multiple choice

A hypothetical town wants to know how residents get to work. Which collection method is most likely to give biased results for the whole town?

- A. Mailing the question to a randomly chosen set of addresses across every neighborhood.
- B. Asking only people waiting at one train station at 8 a.m.
- C. Asking the question on every form residents fill out when they renew a town ID, no matter where they live.
- D. Calling a random selection of phone numbers from across the town at different times of day.

4.1.5 Incomplete and Inaccurate Data

Even when the collection method is fair, the data set itself can be flawed. Real data is messy: a sensor turns off, someone skips a question, a value is typed wrong.

A data set is **incomplete** when some values are missing. It is **inaccurate** when some values are wrong. Either way, a program that uses the data can work **incorrectly** (producing wrong results) or **inefficiently** (doing extra, wasted work). As in Topic 3.2, the program may have no logic error of its own; the trouble comes in with the data.

KEY IDEA

Some data sets are incomplete or contain inaccurate values. A program built or run on such data can give wrong results or waste work, even when its code is correct.

WORKED EXAMPLE A step tracker with a missing day

A hypothetical fitness program averages a week of daily step counts. On day 3 the tracker's battery died, and the app recorded 0 for that day instead of marking it as missing.

StepAverage.java

```

1  import java.util.Scanner;
2
3  public class StepAverage
4  {
5      public static void main(String[] args)
6      {
7          Scanner readings = new Scanner("8200 7900 0 8400 8100");
8          int total = 0;
9          int days = 0;
10         for (int reading = 1; reading <= 5; reading++)
11         {
12             int steps = readings.nextInt();
13             total += steps;
14             days++;
15         }
16         System.out.println("Days counted: " + days);
17         System.out.println("Average steps: " + total / days);
18     }
19 }

```

Days counted: 5

Average steps: 6520

Value read	total	days
8200	8200	1
7900	16100	2
0	16100	3
8400	24500	4
8100	32600	5

32600 / 5 is 6520. But the person actually walked about 8,000 steps every day they were measured. The 0 is not a real day of zero steps; it is a missing value disguised as data. The loop added it like any other value and counted a fifth day, dragging the average far down. The code is correct, and the result is still wrong.

Now vary how the missing day is recorded. If the tracker records a missing reading as -1, a value no real step count can have, the program can recognize and skip it:

StepAverageSkipMissing.java

```

1  import java.util.Scanner;
2
3  public class StepAverageSkipMissing
4  {
5      public static void main(String[] args)
6      {
7          Scanner readings = new Scanner("8200 7900 -1 8400 8100");
8          int total = 0;
9          int days = 0;
10         for (int reading = 1; reading <= 5; reading++)
11         {
12             int steps = readings.nextInt();
13             if (steps >= 0)
14             {
15                 total += steps;
16                 days++;
17             }
18         }
19         System.out.println("Days counted: " + days);
20         System.out.println("Average steps: " + total / days);
21     }
22 }

```

```

Days counted: 4
Average steps: 8150

```

The `if` on line 13 adds only values that are `0` or more, so the `-1` is skipped and only four days count: $32600 / 4$ is 8150 . Notice what made the fix possible: the data marked the missing value clearly. When a missing value looks exactly like a real one, as the `0` did, no program can tell them apart. Complete, accurate data starts at collection time.

Flawed data can also waste work. If a contact list contains each person three times because of repeated imports, a reminder program sends three identical messages to everyone and does three times the work it needed to. The program runs correctly; the data makes it inefficient.

WATCH OUT

"The program ran without errors, so the result must be right." A program that compiles and finishes tells you only that the code followed Java's rules. It says nothing about whether the values it read were complete and correct. Check the data, too: look for missing values, impossible values, and duplicates.

PAUSE AND TRY**Question 8** Multiple choice

A hypothetical weather program averages daily rainfall. Which problem is an example of **inaccurate** data rather than incomplete data?

- A. The sensor was offline for two days, so those days have no reading.
- B. A reading of 15 centimeters was typed as 150 centimeters.

- C. The program uses integer division, so the average loses its decimal part.
- D. Nobody recorded the rainfall on weekends.

4.1.6 Choosing a Data Set That Fits the Question

The last check comes before any code is written: is this data set even the right one? A data set can be fairly collected, complete, and accurate, and still be the wrong tool for your question.

Data is usually gathered to answer one particular question or to describe one particular topic. Its contents reflect that purpose: what was measured, about whom, when, and where. If you try to use it to answer a **different** question, it may simply not contain the information needed, and any answer or prediction you pull out of it may be wrong.

KEY IDEA

A data set is gathered for a specific question or topic. It might not give correct answers, or support correct predictions, for a different question.

WORKED EXAMPLE One data set, two questions

A hypothetical school has a data set of every book checked out of its library last year: book title, date, and the grade level of the borrower.

Question	Does this data set fit?	Why
Which books were checked out most often last year?	Yes	That is exactly what the data records
Which months are busiest for checkouts?	Yes	Every checkout has a date
How many books do students read per year, counting all their reading?	No	Books from home, other libraries, or e-readers never appear in this data
Which books do students enjoy most?	Not reliably	A checkout shows a book was borrowed, not that it was finished or liked

For the first two questions, the data set was gathered about exactly the right thing. For the last two, the data is about something nearby but different, so an answer drawn from it would look precise and still be wrong.

Now vary the question in time instead of topic. Suppose the school wants to predict checkouts for next year, but the library is moving to a new building twice the size, near the cafeteria. Last year's data was collected under very different conditions, so extrapolating from it is risky. The data is the same; the question changed, and the data no longer fits it well.

WATCH OUT

"We already have data, so let's use it." It is tempting to answer a question with whatever data is on hand. Start from the question instead: what information would answer it, and does

this data set actually contain that information? If not, the honest answer is "we need different data."

PAUSE AND TRY

Question 9 Multiple choice

A hypothetical city has a data set of every bus ride last year, recording the route and the time the rider boarded. Which question can this data set answer best?

- A. Which bus routes were busiest at 8 a.m. last year?
- B. Why some residents choose to drive instead of taking the bus.
- C. How many residents own bicycles.
- D. Which new route people would use most if the city added one.

Question 10 Short answer

A hypothetical team has a data set of scores from an online math game played by volunteers ages 10 to 12. A classmate wants to use it to decide how hard to make a math test for high school seniors. Explain why the data set is not appropriate.

Putting it together

A hypothetical student, Priya, is building a program that suggests a study partner to each student in her school. Follow her through every idea in this chapter.

Privacy (4.1.1, 4.1.2). Her first draft of the sign-up form asks for name, email, classes, home address, and phone location. She asks of each field, "does the program need this?" Matching study partners needs names, classes, and a way to contact them, so she keeps those and drops the address and location. Her confirmation screen shows the email as `p***@example.com` instead of the full address, and she plans to delete the data at the end of the school year.

Collection method (4.1.4). To decide which subjects to support first, she posts a survey on the computer science club's page. Most responses ask for help with programming. She realizes the survey only reached club members, who are exactly the students most interested in programming, so she sends it to every homeroom before deciding.

Incomplete and inaccurate data (4.1.5). Some students left "free period" blank, and her first version treated a blank as "no free periods," so those students were never matched. She changes the form to require the field and has the program report anyone whose data is missing instead of silently skipping them.

Algorithmic bias (4.1.3). In testing, students who take classes at a nearby college are never matched, every time, because her rule only compares classes from the school's own course list. The error is systemic, repeated, and always hurts the same group, so it is algorithmic bias. She fixes the rule so it compares free periods even when a class is not on the list.

Fitness of the data set (4.1.6). A teacher offers last year's study-hall attendance as "data about who studies together." Priya notes that attendance shows who was in the same room, not who studied well together, so she does not use it to rank matches.

At each step, the code mattered less than the questions Priya asked about the data: should I collect it, how was it gathered, is it complete and correct, is my rule fair, and does it answer my question?

Vocabulary

Term	Definition	Example
data set	A collection of related pieces of information	All the club sign-up forms
personal data	Information about a specific person	A name, address, or location
privacy	A person's ability to control who knows information about them	Choosing who sees your phone number
safeguard	Protect; for privacy, design a program so users' personal data is not exposed or misused	Showing only the last four digits
algorithmic bias	Errors built into a program that happen again and again and treat one group of users unfairly	Rejecting every two-letter last name
systemic	Built into how a program or its data works, not caused by chance	A rule that always applies
collection method	How a data set was gathered: who, where, when, and how	A survey taken only in the gym
bias (in data)	A tilt in how data was collected that over- or under-represents some people or situations	Only gym users answered
extrapolate	Use data to predict values or reach conclusions beyond what was directly measured	Applying 40 answers to the whole school
incomplete data	A data set with missing values	A day with no step reading
inaccurate data	A data set containing wrong values	81000 typed for 8100

Chapter summary

- Collecting and storing **personal data** puts **privacy** at risk, even in a program with no bugs, and harmless-looking pieces can identify someone when combined.
- Programmers should **safeguard** users' privacy: collect only what is needed, use it only for its purpose, show as little as possible, and delete it when done.
- **Algorithmic bias** is a systemic, repeated error that creates unfair outcomes for a specific group; it can come from a program's rules or from its data.
- Before drawing conclusions or **extrapolating**, check the **collection method** for **bias**; correct code cannot fix data gathered from the wrong people.
- **Incomplete** or **inaccurate** data makes a program work incorrectly or inefficiently, and a data set gathered for one question may not answer a different one.

Review questions

Question 11 Multiple choice

A hypothetical school fitness app records each student's heart rate during gym class. The company that made it plans to share the records with a business that sells health products, without telling students. Which statement is most accurate?

- A. This is fine, because heart rates are just numbers.
- B. This is algorithmic bias, because it affects students.
- C. This is a privacy problem: the personal data is being used for a purpose students never agreed to.
- D. This is inaccurate data, because heart rates change during class.

Question 12 Tracing

The step-average program from 4.1.5 is run on a different week. This time no day is missing, but the value 8100 was mistyped as 81000. What does the program print, and what kind of data problem caused the result?

StepAverageTypo.java

```
1 import java.util.Scanner;
2
3 public class StepAverageTypo
4 {
5     public static void main(String[] args)
6     {
7         Scanner readings = new Scanner("8200 7900 8400 81000");
8         int total = 0;
9         int days = 0;
10        for (int reading = 1; reading <= 4; reading++)
11        {
12            int steps = readings.nextInt();
13            total += steps;
14            days++;
15        }
16        System.out.println("Days counted: " + days);
17        System.out.println("Average steps: " + total / days);
18    }
19 }
```

Question 13 Short answer

A hypothetical company builds a program to pick which job applicants get interviews. It learns what a "good applicant" looks like from the records of people the company hired over the past ten years, almost all of whom came from the same three colleges. Explain how this could lead to algorithmic bias, and connect your answer to how the data was collected.

Question 14 Debugging

A hypothetical library program is supposed to hide most of a library card number and show only the **last** four digits so a patron can recognize their card.

```
1 String libraryCard = "29140068";
2 String shown = "****" + libraryCard;
3 System.out.println("Card: " + shown);
```

Card: ****29140068

What is wrong with line 2 from a privacy point of view, and how should it be corrected?

Question 15 Multiple choice

A hypothetical town measured traffic on Main Street every weekday morning in March, using a sensor that worked every day. A planner wants to use this data to decide how many parking spaces a stadium needs for Saturday night games. Which is the best reason the data set is a poor choice?

- A. The data is incomplete, because the sensor missed days.
- B. The data was collected for a different question: weekday morning street traffic, not Saturday night stadium parking.
- C. The data contains personal data, so it cannot be used.
- D. Traffic data can never be used to make decisions.

Question 16 Short answer

A hypothetical app helps students find tutors. Its sign-up form asks for full name, school, grade, subjects needed, and home address, and it recommends tutors based on ratings left only by students who paid for premium accounts. Identify **two** different issues from this chapter and suggest one change for each.

Reflection

Check each statement. If the answer is "not yet," reread the linked subsection.

- ☐ I can explain why collecting and storing personal data puts privacy at risk. Reread: 4.1.1 Personal Data and Privacy Risks
- ☐ I can describe ways a programmer can safeguard users' privacy. Reread: 4.1.2 Safeguarding Users' Privacy
- ☐ I can define algorithmic bias and tell it apart from a one-time bug. Reread: 4.1.3 Algorithmic Bias
- ☐ I can explain how a biased collection method can make conclusions wrong. Reread: 4.1.4 How the Data Was Collected
- ☐ I can explain how incomplete or inaccurate data makes a program work incorrectly or inefficiently. Reread: 4.1.5 Incomplete and Inaccurate Data
- ☐ I can decide whether a data set fits the question I want to answer. Reread: 4.1.6 Choosing a Data Set That Fits the Question

Check Your Reasoning

Answers and explanations for every question, numbered to match.

Question 1

C. The app holds personal data that, combined, tells exactly where a named student is and when they are alone. If that data is exposed, the student's safety is at risk.

- **A** is wrong: privacy risk comes from holding the data, not from bugs. A perfect program still stores the data.
- **B** is wrong: a schedule alone might be harmless, but combined with a name and home-alone times it reveals much more.
- **D** is wrong: speed is not a privacy issue. The question is who could learn what about the student.

If you chose differently, reread *4.1.1 Personal Data and Privacy Risks*.

Question 2

Model answer: Most runs start and end at the runner's home. Anyone looking at many of the user's routes can see that they all begin and end at the same spot, which reveals the home address even though the app never listed it.

A complete answer must:

- explain that the route data, combined or repeated, points to one location, and
- connect that location to the person's home.

If you thought hiding the address field was enough, reread *4.1.1 Personal Data and Privacy Risks*, especially the part about combining data.

Question 3

The output is:

```
Locker assigned to student ****7325
```

`studentId` has 8 characters (indexes 0 to 7). `substring(4)` returns the characters from index 4 to the end: "7325". The program puts "****" in front, hiding the first four digits so a passerby cannot read the whole ID.

If you got ****4091, you took the characters before index 4 instead of from index 4 on. Reread *4.1.2 Safeguarding Users' Privacy*.

Question 4

B. To show how busy the library is, the app only needs a count. Not collecting names at all means there is nothing personal to expose.

- **A** is wrong: names and times are not needed for a busy meter, and keeping them forever "just in case" adds risk for no benefit.
- **C** is wrong: tracking location all day collects far more than the purpose requires.
- **D** is wrong: `private` limits which code can use the variable; it does not reduce what is collected or who can see the stored data.

If you chose differently, reread *4.1.2 Safeguarding Users' Privacy*.

Question 5

B. The failure is systemic (it comes from how the feature was built), repeated, and lands on a specific group of users. That matches every part of the definition.

- **A** is wrong: a one-time crash caused by the power is not systemic and does not target a group.
- **C** is wrong: a typo is a mistake, but it does not create an unfair outcome for any group.
- **D** is wrong: one unwanted recommendation for one user is not a repeated, unfair outcome for a group.

If you chose differently, reread *4.1.3 Algorithmic Bias*.

Question 6

`indexOf(" ")` returns `-1` only when the name has no space, so every multi-word last name, such as "De La Cruz" or "Van Dyke", makes the condition `false` and is rejected. The rejection is built into the rule (systemic), happens every time (repeated), and always falls on people whose family names have more than one word (a specific group). That is algorithmic bias.

The corrected line is:

```
1 if (lastName.length() > 0)
```

With that change, all four names print `accepted`, and an empty entry would still be rejected. The original program never crashed; it was a fairness error, not a syntax or run-time error. Reread *4.1.3 Algorithmic Bias* if your explanation did not mention all three parts of the definition.

Question 7

B. People waiting at a train station are, almost by definition, people who take the train. The method reaches one kind of commuter and misses drivers, walkers, cyclists, and people who work at home.

- **A** is wrong: random addresses from every neighborhood reach many kinds of residents.
- **C** is wrong: renewal forms go to residents all over town, regardless of how they commute.
- **D** is wrong: random calls at different times reach people with different schedules, which reduces bias. (Any method can have some bias, but B is clearly the most tilted.)

If you chose differently, reread *4.1.4 How the Data Was Collected*.

Question 8

B. The value exists but is wrong, which is inaccurate data. An average using 150 would be far too high.

- **A** is wrong: missing readings make the data incomplete, not inaccurate.
- **C** is wrong: that is a choice in the code, not a flaw in the data set.
- **D** is wrong: unrecorded weekend days are missing values, so the data is incomplete.

If you chose differently, reread *4.1.5 Incomplete and Inaccurate Data*.

Question 9

A. The data records the route and boarding time of every ride, which is exactly what is needed to count rides per route at 8 a.m.

- **B** is wrong: people who drive never appear in bus-ride data, and the data records no reasons.

- **C** is wrong: bicycle ownership is a different topic that this data never measured.
- **D** is wrong: a route that does not exist has no rides recorded, so any prediction about it is an extrapolation the data cannot support well.

If you chose differently, reread *4.1.6 Choosing a Data Set That Fits the Question*.

Question 10

Model answer: The data describes how 10- to 12-year-olds did on a game, not how high school seniors do on a test. Seniors know different math, and a game is different from a test, so the data does not contain the information the question needs. Using it would extrapolate to a group and a task it never measured.

A complete answer must:

- name the mismatch in who was measured (younger students, not seniors), what was measured (a game, not a test), or both, and
- conclude that answers drawn from the data could be wrong for this question.

Reread *4.1.6 Choosing a Data Set That Fits the Question* if you missed the mismatch.

Question 11

C. Heart-rate records tied to students are personal (health) data. Sharing them with a business for a purpose students were never told about breaks the habit of using data only for its stated purpose.

- **A** is wrong: numbers attached to a specific person are personal data, and health data can be sensitive.
- **B** is wrong: algorithmic bias is a repeated error that treats one group unfairly. Nothing here is an error in the program's results.
- **D** is wrong: a heart rate that changes during exercise is accurate; it really did change.

Reread *4.1.1 Personal Data and Privacy Risks* and *4.1.2 Safeguarding Users' Privacy* if you chose differently.

Question 12

Value read	total	days
8200	8200	1
7900	16100	2
8400	24500	3
81000	105500	4

The output is:

```
Days counted: 4
Average steps: 26375
```

105500 / 4 is 26375. With the correct value the total would be 32600 and the average 8150. One wrong value more than tripled the result. This is **inaccurate** data: no value is missing, but one is wrong. The code is correct; the data made the program produce a wrong answer.

Reread *4.1.5 Incomplete and Inaccurate Data* if you called this incomplete data or got a different average.

Question 13

Model answer: The data was collected only from past hires, and past hiring mostly chose people from three colleges. The program learns that "good applicants" look like those people, so it keeps scoring applicants from other colleges lower. That is a systemic, repeated error (it comes from the data the program relies on and happens to every such applicant) that creates an unfair outcome for a specific group: qualified applicants from other colleges. Because of the biased collection, the program extrapolates "who will be good" from a group that never included many kinds of people.

A complete answer must:

- identify the collection method (past hires from mostly three colleges) as biased, and
- explain that this leads to repeated unfair outcomes for a specific group (applicants from other colleges).

Reread *4.1.3 Algorithmic Bias* and *4.1.4 How the Data Was Collected* if your answer missed either connection.

Question 14

Line 2 puts "****" in front of the **whole** card number instead of in place of part of it. The stars hide nothing: anyone who sees the screen reads all eight digits. The program runs without errors, so nothing warns the programmer; the mistake is about how much personal data is shown.

The corrected line keeps only the characters from index 4 on:

```
1 String shown = "****" + libraryCard.substring(4);
```

`libraryCard` has 8 characters, so `substring(4)` is `"0068"`, and the program prints:

```
Card: ****0068
```

Reread *4.1.2 Safeguarding Users' Privacy* if you were unsure which part of the string to keep.

Question 15

B. The data set fits questions about weekday morning traffic on Main Street. Saturday night games bring different people, at a different time, to a different place, so the data does not contain what the planner needs.

- **A** is wrong: the question says the sensor worked every day, so nothing is missing.
- **C** is wrong: car counts on a street are not information about a specific person.
- **D** is wrong: traffic data is very useful for questions it was collected to answer.

Reread *4.1.6 Choosing a Data Set That Fits the Question* if you chose differently.

Question 16

Model answer:

1. **Privacy:** the app collects home addresses, which it does not need to match students with tutors. Change: remove the address field (or ask only for a general area if distance matters).
2. **Biased collection method:** ratings come only from premium users, who may differ from other students (for example, they can afford more sessions). Recommendations built on those ratings may not fit most students. Change: collect ratings from all users, or show how many and which kinds of students left them.

Other good answers include algorithmic bias (tutors popular with premium users get recommended over and over, while others are never shown) or data fit (ratings of tutors in one subject used to recommend tutors in another).

A complete answer must name two different issues, explain each in terms of this app, and give a matching change for each.

Reread *4.1.2 Safeguarding Users' Privacy* and *4.1.4 How the Data Was Collected* if you found only one issue.