

Solutions

Each answer shows the steps, then the **result**.

2.1

```
for (int i = 0; i < words.size(); i++)
{
    System.out.println(words.get(i));
}
```

- loop bound `i < words.size()`
- prints `words.get(i)`

2.2

```
int total = 0;
for (int sc : scores) total += sc;
```

- `total` set to 0 before the loop
- enhanced loop adds each element

2.3

- a `ConcurrentModificationException` is thrown

2.4

- the second 2 moves into index 1 after the first is removed, then index 2 is checked:
[1, 2, 3]

3.1 B

- $(6 - 3) + (9 - 6) = 6$

3.2 C

- the second `tbd` moves into the index just checked and is skipped: [a, `tbd`]

3.3 C

- `n` is fixed at 3 before the loop, so it adds exactly three elements

3.4

(a)

```
public static int countStarting(ArrayList<String> loans, String letter)
{
    int count = 0;
    for (String t : loans)
    {
        if (t.substring(0, 1).equals(letter)) count++;
    }
    return count;
}
```

- traverses every title
- compares the **first character** with `equals`
- counts and **returns** the count

(b)

```
public static void removeReturned(ArrayList<String> loans,
    ↪ ArrayList<String> returned)
{
    for (int i = loans.size() - 1; i >= 0; i--)
    {
        if (returned.contains(loans.get(i)))
        {
            loans.remove(i);
        }
    }
}
```

- indexed traversal of `loans`
- tests `returned.contains(loans.get(i))`
- `remove(i)` on `loans`
- traverses **backwards** (or only increments when nothing is removed), so nothing is skipped