Solutions

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

2.1

- the data and the methods that act on it are bundled in one class
- the data is hidden and can only be reached through those methods

2.2

- the mutator can reject invalid values, so the object stays valid
- the inside of the class can change without breaking code that uses it

2.3

- name compartment `Member`
- `- name: String`, `- credits: int`
- `+ Member(name)`, `+ add_credits(n): void`, `+ book(): bool`, `+ get_credits(): int`
- `-` on the attributes and `+` on the methods

2.4

- **0.0**: the constructor stores `max(0.0, -20.0)`

3.1 B

- `-` private, `+` public

3.2 B

- it returns a value and changes nothing

3.3 B

- check, then change

3.4

(a)

- name compartment `Locker`
- private attributes with types: `- number: int`, `- locked: bool`, `- code: String`
- constructor `+ Locker(number, code)`
- `+ lock(): void`, `+ unlock(code): bool`, `+ is_locked(): bool`

(b) One answer:

```
def unlock(self, code):
    if code == self._code:
        self._locked = False
        return True
    return False
```

- compares `code` with the stored code
- sets `locked` to `False` only on a match
- returns `True` or `False`

(c)

- other code cannot read the code and open any locker
- the only way in is through `unlock`, which checks the code