Solutions

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

2.1

- (i) `"B4"`: the constructor is inherited
- (ii) no lounge: **18**

2.2

- (i) **composition**
- (ii) **inheritance**: a teacher is an employee
- (iii) **aggregation**

2.3

- overriding: a subclass redefines a superclass method with the same name and parameters
- overloading: methods in one class share a name but have different parameters

3.1 B

- `class Bus extends Vehicle`

3.2 B

- an abstract method must be overridden

3.3 B

- the inherited attributes are set in one place

3.4

(a)

- van: $900 + 250 = 1150 \leq 1200$: **True**
- bike: $30 + 15 = 45 > 40$: **False**

(b)

- `can_add` calls `self.max_load()`
- each subclass overrides `max_load`, so the object's own version runs: polymorphism

(c) One answer:

```
class Truck(Vehicle):
    def __init__(self, reg, load, trailer):
        super().__init__(reg, load)
        self.trailer = trailer
    def max_load(self):
        if self.trailer:
            return 10000
        return 8000
```

- `class Truck(Vehicle)`
- constructor calls `super().__init__(reg, load)`
- stores `trailer`
- overrides `max_load` with 8000 or 10000

(d)

- **aggregation**
- the vehicles exist on their own and move to another depot when this one closes