1-1:
Rubber tire, lug nuts, metal wheel, car body, engine, mirrors, car doors and windows, windshield
Almost none of them, possibly the windows, wheels, and lug nuts

1-2:
I live at home, but here are some components: Bed, desk, dresser, carpet, door, closet doors, sofa chair

1-3:
Computer chip, Display screen, Keyboard, Speakers, Camera, Microphone, Fan, Casing, Battery, Trackpad
Depending on the company, the computer chip is sometimes assembled by the company, the battery, and casing.

2-1:
To “order from the library”, to bring in pre-assembled parts into scope

2-2:
Primitive types exist in java without the need to import and work with basic operators, while component types must be imported

2-3:
boolean

2-4:
```
if ((0 <= yourAge) && (yourAge <= 3)) {
        output.println("My, just 2 years old!");
        output.println("What a cute little baby.!");
}
else {
        output.println("Thanks for entering your age.");
}
```

2-5:
In all cases:
“Oh no! A teenager!
So sorry!

3-1:
Macbook Pro
Airplane

3-2:
In general, people read instruction manuals etc before trying to use something, so thgey try to figure out as many details as possible before using it. In software, you’re never going to be able

to figure out ALL the details, but the most important part is that people who use software components need to learn what to do to use something, not necessarily be bogged down with all the details.

3-3:
F = ma models force in relation to mass and acceleration
y = x^2 models projectiles

3-4:
Because you can manipulate models to test and substitue different values and simluate and manipulate outcomes

3-5:
13, -5

3-6:
All of the constraints are provided in the AMPMClock type, and there are no further constraints from the type to the model

4-1:
clock.display(), clock.daylightSavings(), clock.plusOneHour(), clock.minusOneHour(), clock.reset()

4-2:
The formal parameters and description of AMPMClock

4-3:
Incoming values, because the outgoing values of paramters are not copied back to the arguments after the method executes

4-4:
newHours = 3
myClock = (3,25,48,true)

4-5:
void setSeconds(int newSeconds)
Sets this.seconds to newSeconds
Parameters: newSeconds – the new seconds for this
Updates: this.seconds
Requires: 0 <= newSeconds <= 59
Ensures: this.seconds = newSeconds

4-6:
newMinutes will equal 31 after both method calls

4-7:
myClock = (11,25,48,true)

4-8:
boolean, because isAm returns this.am, which is a boolean

4-9:
the value that isAm returns is either true or false - whether clock's am value is true or false

4-10:
The value of clock

4-11:
int minutes()
Reports this.minutes.
Returns: this.mimutes
Ensures: minutes = this.minutes

4-12:
int seconds()
Reports this.seconds
Returns: this.seconds
Ensures: seconds = this.seconds

4-13:
myClock = (8,2,43,true)
yourClock = (11,18,6,false)
transferMinutes = 2

myClock = (8,2,43,true)
yourClock = (11,2,6,false)
transferMinutes = 2