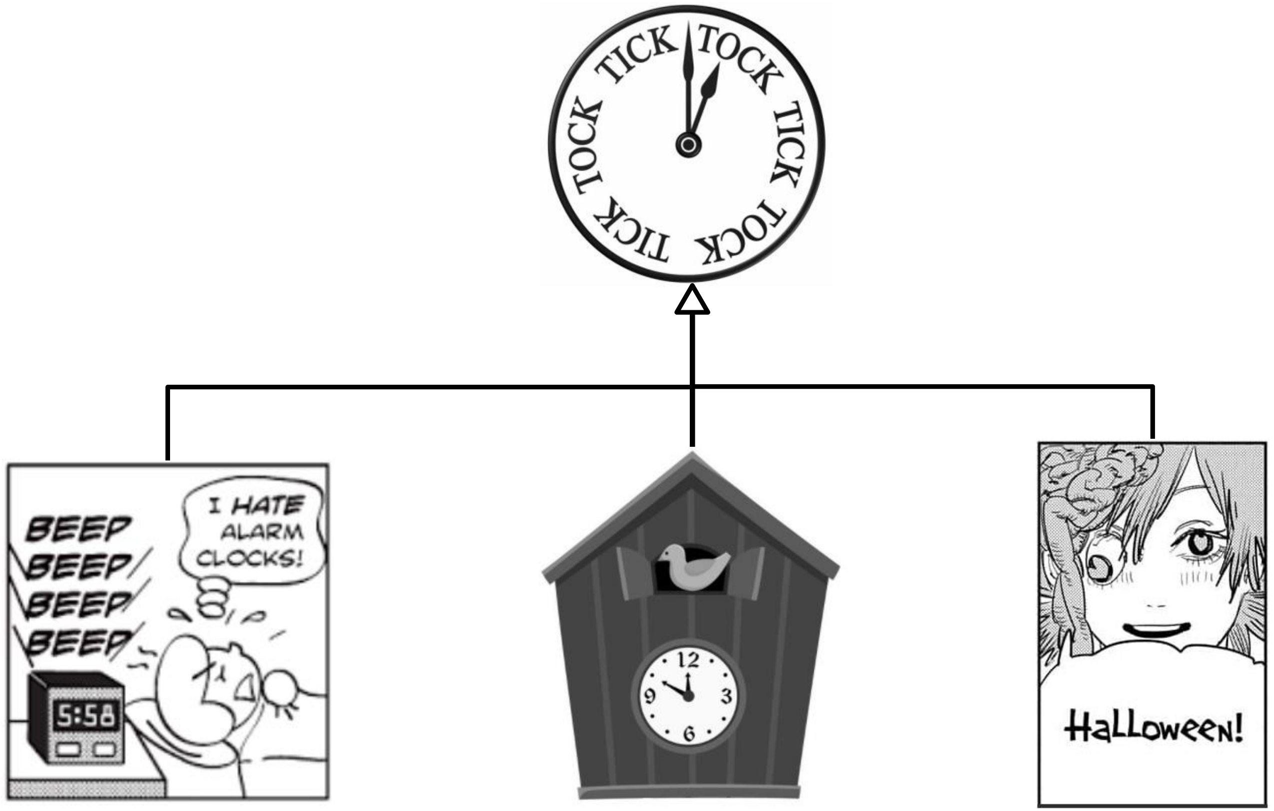


Java Programming

CPT111 – Week 9

Erick Purwanto

Xi'an Jiaotong-Liverpool University
西交利物浦大学

CPT111 Java Programming

Week 9 Exercise and Coursework 1

More Objects and Inheritance

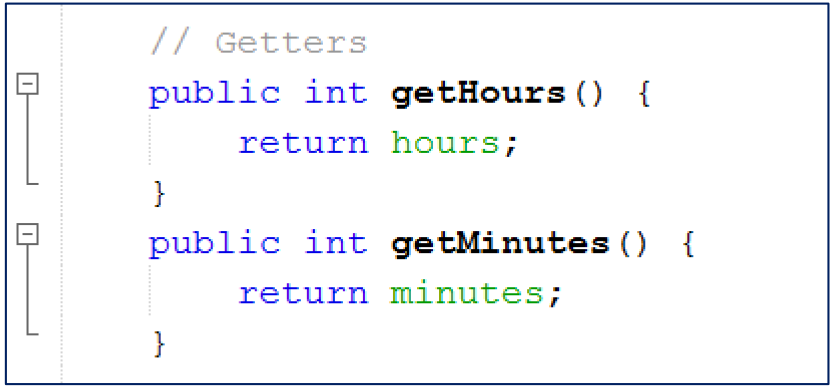
Coding and Submission

- Coding in your NetBeans
  - Start with the skeleton code given in the course LMO

- Submitting into Learning Mall Quiz
  - Do not submit the whole class
  - Only submit the constructor or the method
    - read carefully the instructions
  - You can submit your own private helper method
    - but *do not* add another public methods

Clock + Getters

- In the exercises and CW1s of Week 9
  - we will subclass the class Clock from Week 7
  - but with **<u>you must add these getters into it</u>**



```
// Getters
public int getHours() {
    return hours;
}
public int getMinutes() {
    return minutes;
}
```

  - complete Clock.java with your Week 7 code or the standard solution code!

Alarm Clock

- AlarmClock is a subclass of Clock
  - you can still use the inherited methods such as toString()

- In addition, you can now also set up an alarm to alert you in a specified time
  - using the default or a chosen message
  - by overriding Clock's method tick()

Exercise #9.1 AlarmClock Constructor 1

- Complete the first constructor of the class AlarmClock
- It takes four arguments: h, m, alarmHours, alarmMinutes
  - creates a new AlarmClock object whose initial time is h hours and m minutes
  - sounds an alarm at alarmHours hours and alarmMinutes minutes, with the default sound "Beep beep beep beep!"

- Test cases:

```
AlarmClock ac1 = new AlarmClock(5, 58, 6, 0);
ac1.tick();
ac1.tick();                              → Beep beep beep beep!
System.out.println(ac1);                 → 06:00
```

Exercise #9.2 AlarmClock Constructor 2

- Complete the second constructor of the class AlarmClock
- Overloading the first constructor, it now takes five arguments: h, m, alarmHours, alarmMinutes, and alarmSound
  - creates a new AlarmClock object whose initial time is h hours and m minutes
  - sounds an alarm at alarmHours hours and alarmMinutes minutes, and sets the sound to alarmSound
- Test cases:

```
AlarmClock ac2 = new AlarmClock(14, 29, 14, 30, "Wake Up! The Hero! Kamen Rider!");
ac2.tick();                              → Wake Up! The Hero! Kamen Rider!
```

Exercise #9.3 AlarmClock Tick

- Complete the method tick of the class AlarmClock
- It overrides the method tick of Clock and adds 1 minute to the time on this alarm clock
  - In addition, it sounds (prints) the alarm at the specified time.

- Test cases:

```
AlarmClock ac1 = new AlarmClock(5, 58, 6, 0);
ac1.tick();
ac1.tick();                          → Beep beep beep beep!
System.out.println(ac1);             → 06:00
AlarmClock ac2 = new AlarmClock(14, 29, 14, 30, "Wake Up! The Hero! Kamen Rider!");
ac2.tick();                          →  Wake Up! The Hero! Kamen Rider!
```

Cuckoo Clock

- CuckooClock is a subclass of Clock
  - you can still use the inherited methods such as toString()

- In addition, the CuckooClock can print "Cuckoo!" at the start of every hour
  - as many as the current hours, no matter it is morning or night

Exercise #9.4 CuckooClock Constructor

- Complete the constructor of the class CuckooClock
- It takes two arguments: h and m
    - and creates a new CuckooClock object whose initial time is h hours and m minutes

- Test cases:

```
CuckooClock cc1 = new CuckooClock(0, 58);
cc1.tick();
cc1.tick();                                              → Cuckoo!
System.out.println(cc1);                                 → 01:00
CuckooClock cc2 = new CuckooClock(13, 59);
cc2.tick();                                              → Cuckoo!↲Cuckoo!
```

CW1 #9.1 CuckooClock Tick

- Complete the method tick of the class CuckooClock
- It overrides the method tick of Clock and adds 1 minute to the time on this Cuckoo clock
- In addition, it prints "Cuckoo!" at the start of every hour
  - It prints one time for each hour
  - Whether it is morning or night does **not** change the number of times it prints

- Test cases:

```
CuckooClock cc1 = new CuckooClock(0, 58);
cc1.tick();
cc1.tick();                                    → Cuckoo!
System.out.println(cc1);                       → 01:00
CuckooClock cc2 = new CuckooClock(13, 59);
cc2.tick();                                    → Cuckoo!↲Cuckoo!
```

Halloween Clock

- HalloweenClock is a subclass of Clock
  - you can still use the inherited methods such as toString()

- In addition, whenever ***any*** current instances of HalloweenClock have ticked three times
  - the latest ticking object will also print "Halloween!"

CW1 #9.2 HalloweenClock Constructor

- Complete the constructor of the class HalloweenClock
- It takes two arguments: h and m
  - and creates a new HalloweenClock object whose initial time is h hours and m minutes
- Test cases:

```
HalloweenClock hc1 = new HalloweenClock(1, 0);
HalloweenClock hc2 = new HalloweenClock(2, 0);
hc1.tick();
hc2.tick();
hc2.tick();                          → Halloween!
HalloweenClock hc3 = new HalloweenClock(3, 30);
hc1.tick();
hc2.tick();
hc3.tick();                          → Halloween!
System.out.println(hc3);       → 03:31
```

CW1 #9.3 HalloweenClock Tick

- Complete the method tick of the class HalloweenClock
- It overrides the method tick of Clock, adds 1 minute to the time on this Halloween clock
  - and if ***any*** Halloween clocks have ticked three times, prints "Halloween!"
- Test cases:

```
HalloweenClock hc1 = new HalloweenClock(1, 0);
HalloweenClock hc2 = new HalloweenClock(2, 0);
hc1.tick();
hc2.tick();
hc2.tick();                    → Halloween!
HalloweenClock hc3 = new HalloweenClock(3, 30);
hc1.tick();
hc2.tick();
hc3.tick();                    → Halloween!
System.out.println(hc3);       → 03:31
```

Thank you for your attention!

- This is the end of Week 9 Exercise and Coursework 1 Task Sheet