# ZombieTime – JSR 310 for the Undead

Stephen Chin (@steveonjava)
Java Technology Evangelist
JavaOne Conference Chair

JavaOne℠
ORACLE

# We just want to fit in

https://www.flickr.com/photos/philippeleroyer/4071842319/

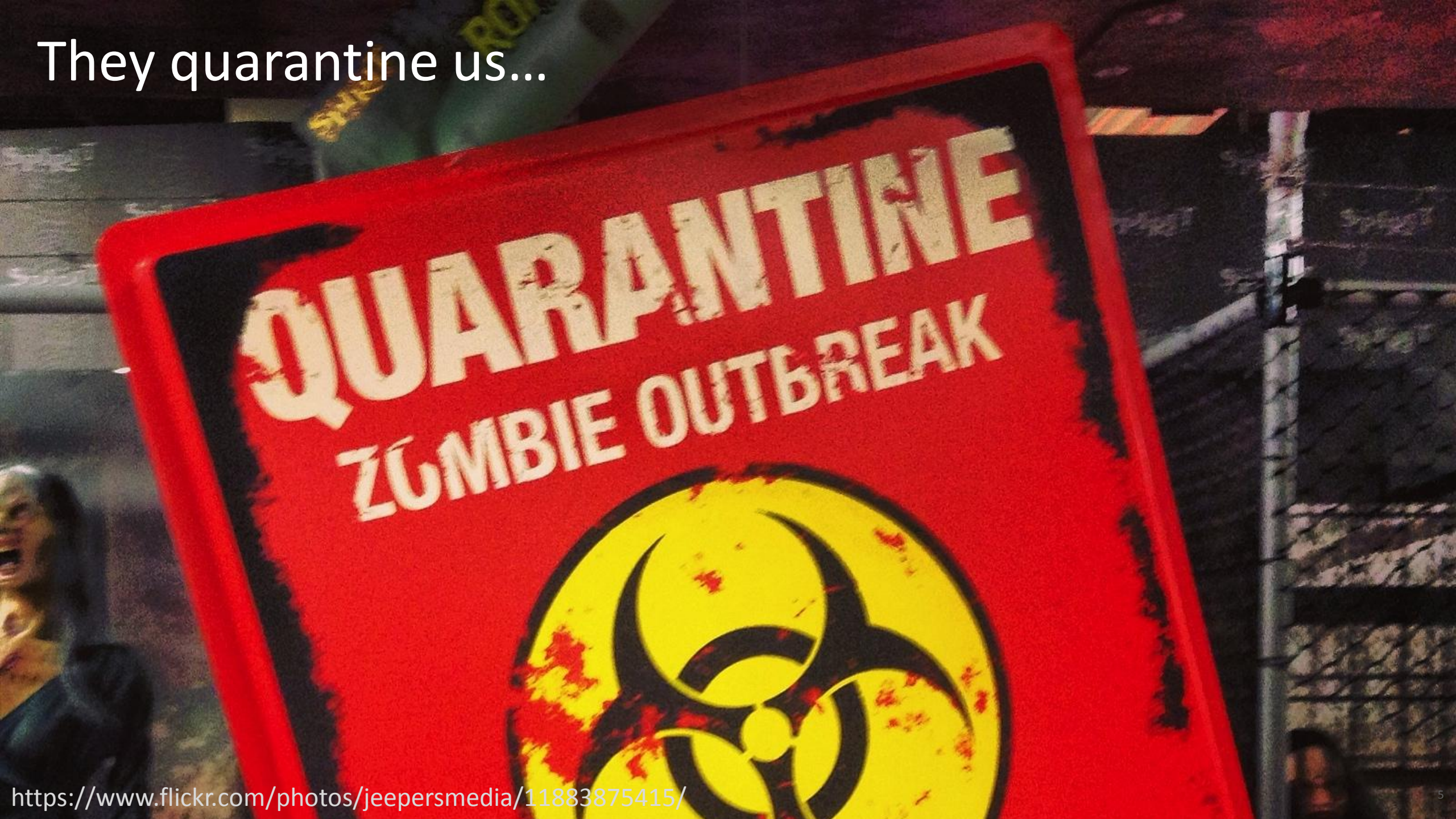
# But humans are scared of us!

https://www.flickr.com/photos/moira_fee/5632351014/

# Is it just because we eat brains?

https://www.flickr.com/photos/dwilliss/7861599590/

# They quarantine us…




https://www.flickr.com/photos/jeepersmedia/11883875415/

# And attack us with weapons…

https://www.flickr.com/photos/nichpics/10628256503/

# But when it is us vs. them…

https://www.flickr.com/photos/icedsoul/4511648609/in/photostream/

# Humans don't stand a chance!

https://www.flickr.com/photos/icedsoul/4470494133/

# JSR 310, a New Weapon Against the Humans!

- Our researchers have come up with a new weapon against the humans!
  - Modern API – Zombies type slow, so we need a fluent API with easy conversions
  - Thread safe – Don't get caught in a deadlock with armed humans!
  - Simplified Time Zone Handling – Coordinated global attacks!


https://www.flickr.com/photos/soul_stealer/8249718101/

# Don't repeat this failure…

- // hit the humans while they are celebrating!
- Date attackDate = new Date(2013, 12, 25);

+1900

0-based (No Range Checking!)

In the yea 3914 3

https://www.flickr.com/photos/frogdna/6948427148/

# January 25th, 3914 : Failed Zombie Trooper Invasion


https://www.flickr.com/photos/starwarsblog/516181007/

# ZombieTime – Undead Trainer for JSR 310

# Basic Concepts

- LocalDate, LocalTime, LocalDateTime
  - Represents a calendar date, time, or both
- Instant
  - Amount of time since the epoch – ideal for timestamps and calculations
- Period
  - A span of time between two dates (goes well with LocalDateTime)
- Duration
  - A precise span of time (goes well with Instants)

# Creating Dates and Times

- Now
  - LocalDate.now()
- Static
  - LocalDate.of(2013, Month.DECEMBER, 25)  // Use the enums!
  - LocalTime.of(11, 30, 5, 999_999_999)  // Time to go to work…
  - LocalDateTime.of(2013, Month.DECEMBER, 25, 11, 30, 5, 999_999_999)  // Even on Christmas
- Parsing
  - LocalDate.parse()
- Conversion
  - new Date().toInstant()
  - Calendar.getInstance().toInstant()

# PixelatedClock

```
Timeline clockTimeline = new Timeline(
  new KeyFrame(Duration.millis(1),
  actionEvent -> {
    LocalTime now = LocalTime.now();
}));
```

ZombieTime

23:03:37.164

# Formatting Dates and Times

- Constants
  - DateTimeFormatter.BASIC_ISO_DATE  // '2011-12-03+01:00'
  - DateTimeFormatter.ISO_TIME // '10:15:30+01:00'
  - DateTimeFormatter.ISO_DATE_TIME // '2011-12-03T10:15:30+01:00[Europe/Paris]'
  - DateTimeFormatter.ISO_WEEK_DATE // 2012-W48-6'
- Localized Formatters
  - DateTimeFormatter.ofLocalizedTime(FormatStyle.SHORT)
- Strings
  - date.toString("d MMM uuuu")  // '25 Dec 2014'
- Formatters are immutable and thread-safe for reuse!

# Localized PixelatedClock

```
DateTimeFormatter clockFormat =
DateTimeFormatter.ofLocalizedTime(FormatStyle.SHORT);


Timeline clockTimeline = new Timeline(
  new KeyFrame(Duration.millis(1),
  actionEvent -> {
    setText(now.format(clockFormat));
}));
```

ZombieTime

9:02 AM

GAME OVER

# I HATE SUNBURN!

https://www.flickr.com/photos/rosenwald/4534092922/

# Using LocalTime/LocalDate

- Comparing Times/Dates:
  - time.isBefore(LocalTime.NOON) // before noon
  - date.isAfter(LocalDate.ofYearDay(2014, 365 / 2)) // appx. halfway through the year
- Year Info:
  - date.getYear()
  - date.getDayOfYear()
  - date.lengthOfYear()
  - date.isLeapYear()
- Time Info:
  - time.getHour() / time.getMinute() / time.getSecond() / time.getNano()

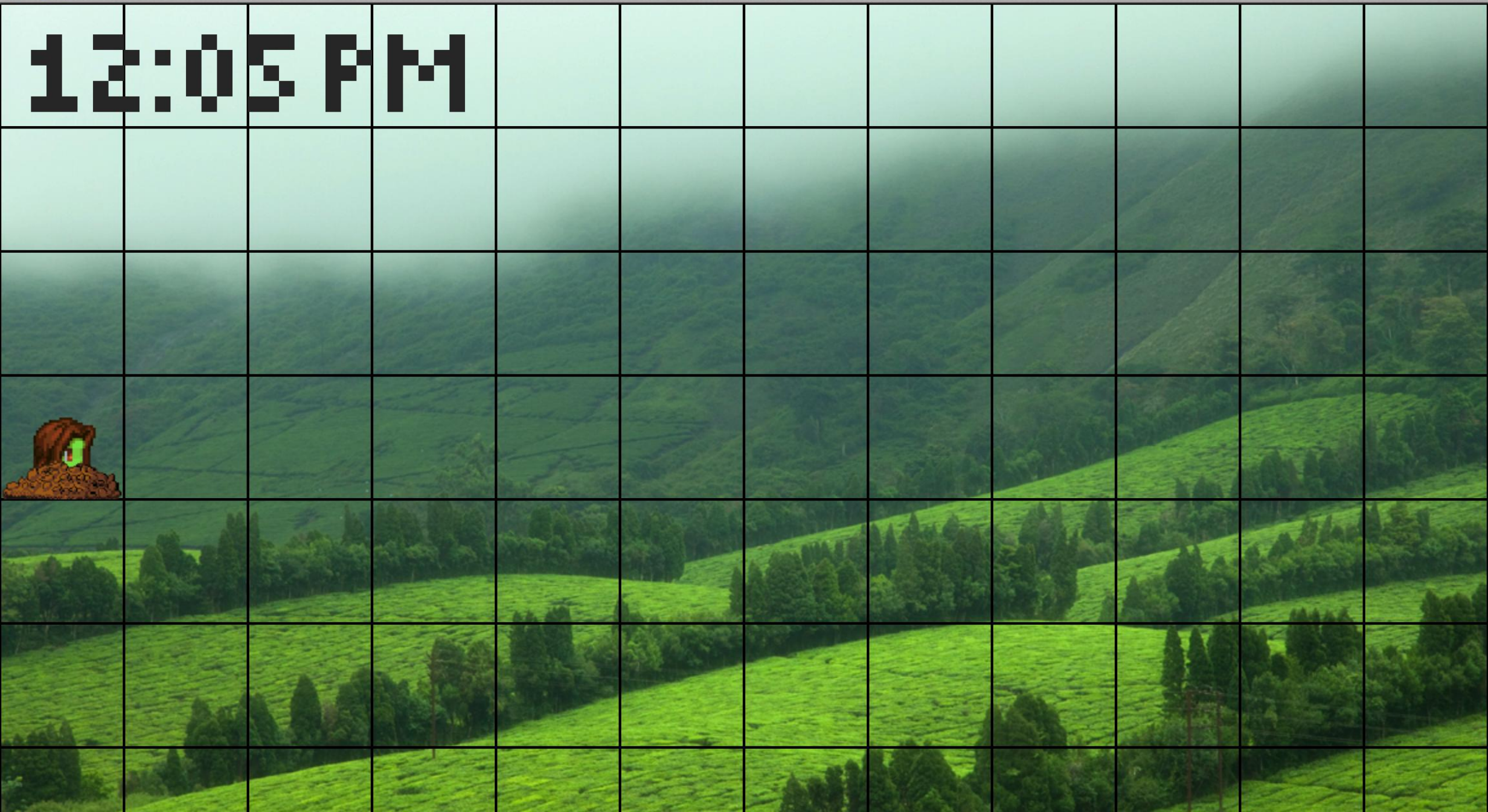
# Hide from the sunlight!

```
public BooleanProperty isNight = new
SimpleBooleanProperty();


isNight.setValue(now.isAfter(LocalTime.of(6, 0)) &&
now.isBefore(LocalTime.of(19, 0)));


underground.bind(Main.pixelatedClock.isNight.not());
```

ZombieTime



# Let's Add Some Villagers!

```
children.add(new SpriteView.Fred(new Location(8, 2)));
children.add(new SpriteView.Sam(new Location(9, 4)));
children.add(new SpriteView.Ted(new Location(7, 6)));
children.add(new SpriteView.Sarah(new Location(5, 4)));
children.add(new SpriteView.Jenn(new Location(6, 5)));
```


8:58 AM
Ouch! Don't step on me

# Using Time Zones

Creating ZoneIds:

- zoneId = ZoneId.of("Europe/Paris");
- zoneId = ZoneId.of(ZoneId.SHORT_IDS.get("EST"));
- zoneId = ZoneOffset.ofHours(-8);

Using ZoneIds:

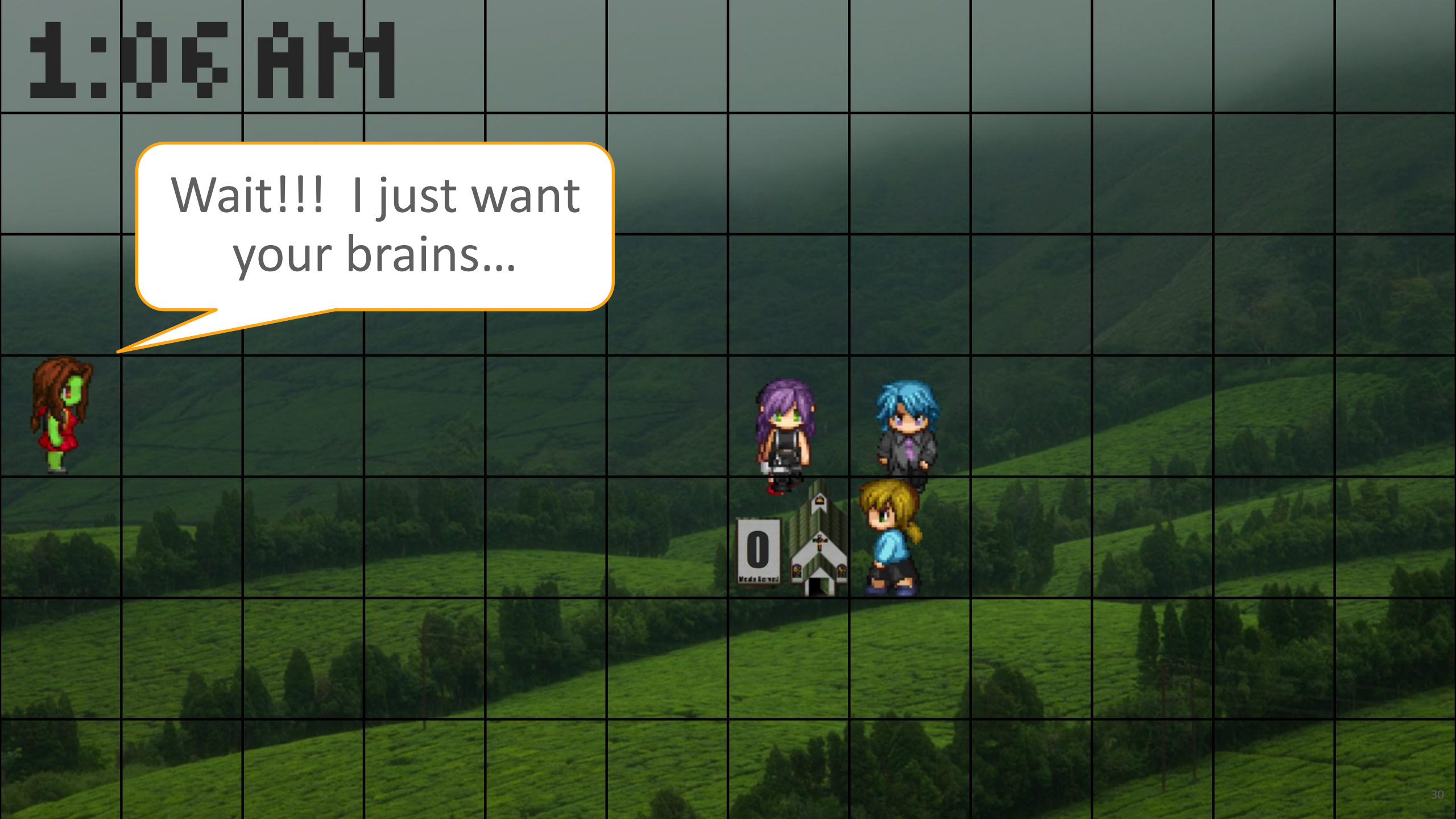
- dateTime.now(zoneId);
- dateTime.atZone(zoneId);
- Clock.system(zoneId);

# Make it Night!

```
ZoneId zoneId = ZoneId.of("Europe/London");
LocalTime now = LocalTime.now(zoneId);
```

# Make it Night! (with a clock)

```
ZoneId zone = ZoneId.of("Europe/London");
Clock clock = Clock.system(zone);
LocalTime now = LocalTime.now(clock);
```

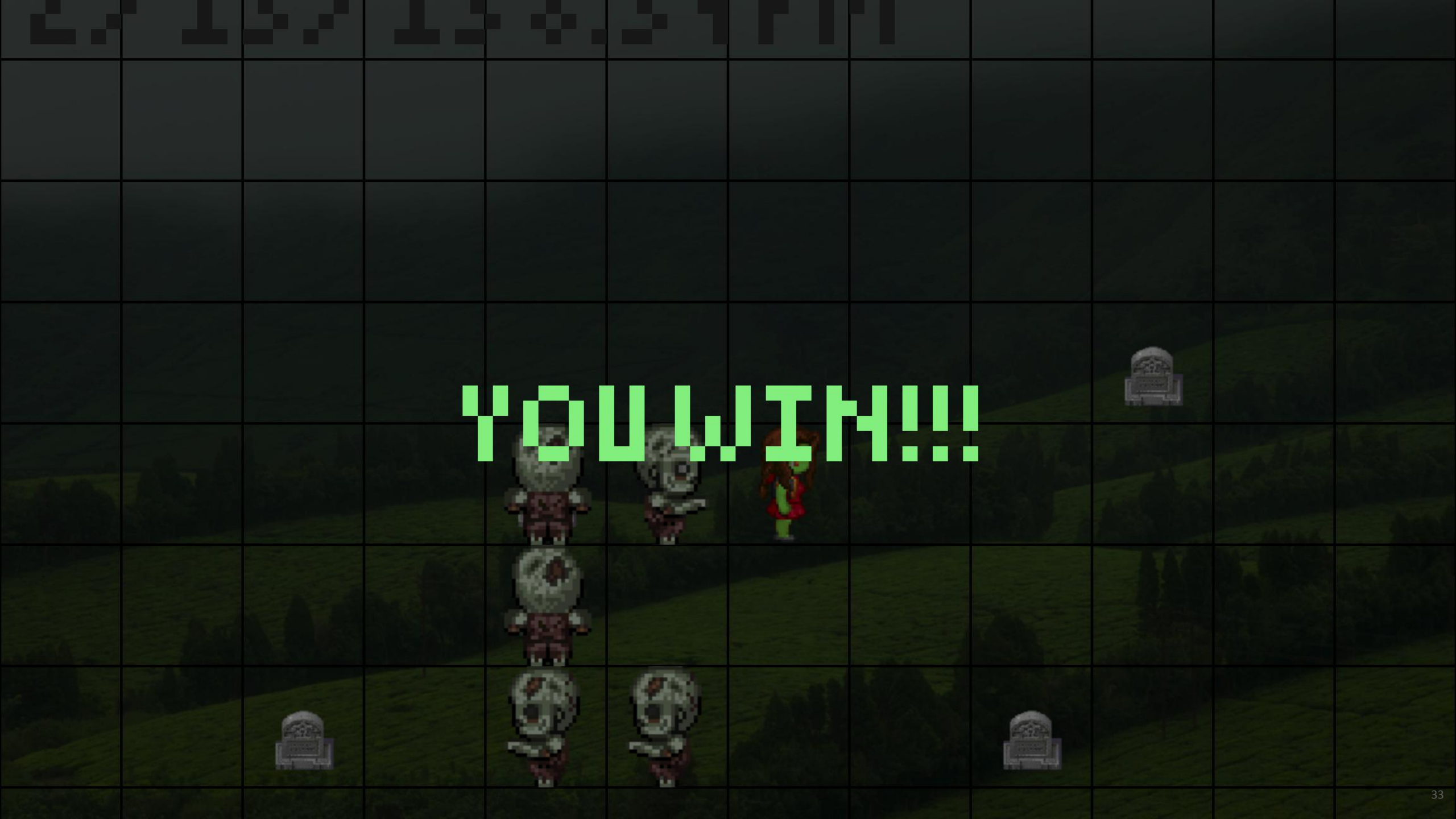

1:06 AM
Wait!!! I just want your brains…

# Temporal Adjustors

- Predefined adjustors for common cases
  - First or last day of month
    - date.with(TemporalAdjusters.firstDayOfMonth())
  - First or last day of year
    - date.with(TemporalAdjusters.firstDayOfNextYear())
  - Last Friday of the month
    - date.with(TemporalAdjusters.lastInMonth(DayOfWeek.FRIDAY))
- Custom adjustors for your own business logic

# Friday 13th TimeAdjustor

```
TemporalAdjuster friday13Adjuster = temporal -> {
    if (temporal.get(ChronoField.DAY_OF_MONTH) > 13) temporal =
        temporal.plus(1, ChronoUnit.MONTHS);
    temporal = temporal.with(ChronoField.DAY_OF_MONTH, 13);
    System.out.println("temporal = " + temporal);
    while (temporal.get(ChronoField.DAY_OF_WEEK) !=
           DayOfWeek.FRIDAY.getValue()) {
        temporal = temporal.plus(1, ChronoUnit.MONTHS);
        System.out.println("temporal = " + temporal);
    }
    return temporal;
};
```


YOU WIN!!!


https://www.flickr.com/photos/s1mone/1833002341


Stephen Chin
tweet: @steveonjava
blog: http://steveonjava.com

NightHacking Tour


Real Geeks
Live Hacking

nighthacking.com

# Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

JavaOne™

ORACLE®