## Instructions

Manage a game player's High Score list.

Your task is to build a high-score component of the classic Frogger game, one of the highest selling and addictive games of all time, and a classic of the arcade era. Your task is to write methods that return the highest score from the list, the last added score and the three highest scores.

Solution

```
class HighScores {

    HighScores() {

        throw new UnsupportedOperationException('method not implemented.')

    }

}



import spock.lang.*



class HighScoresSpec extends Specification {



    def "List of scores"() {

        expect:

        new HighScores(scores).scores == expected



        where:
```

```
    scores           || expected
    [30, 50, 20, 70] || [30, 50, 20, 70]
}


@Ignore
def "Latest score"() {
    expect:
    new HighScores(scores).latest() == expected


    where:
    scores           || expected
    [100, 0, 90, 30] || 30
}


@Ignore
def "Personal best"() {
    expect:
    new HighScores(scores).personalBest() == expected


    where:
    scores         || expected
    [40, 100, 70] || 100
}
```

```
@Ignore
def "Top 3 scores"() {
    expect:
    new HighScores(scores).personalTopThree() == expected

    where:
    scores                                               || expected
    [10, 30, 90, 30, 100, 20, 10, 0, 30, 40, 40, 70, 70] || [100, 90, 70]
}

@Ignore
def "Personal top highest to lowest"() {
    expect:
    new HighScores(scores).personalTopThree() == expected

    where:
    scores       || expected
    [20, 10, 30] || [30, 20, 10]
}

@Ignore
def "Personal top when there is a tie"() {
    expect:
    new HighScores(scores).personalTopThree() == expected

    where:
    scores           || expected
    [40, 20, 40, 30] || [40, 40, 30]
}

@Ignore
def "Personal top when there are less than 3"() {
    expect:
    new HighScores(scores).personalTopThree() == expected

    where:
    scores   || expected
    [30, 70] || [70, 30]
}

@Ignore
def "Personal top when there is only one"() {
    expect:
    new HighScores(scores).personalTopThree() == expected
    where:
    scores || expected
    [40]   || [40]
  }

  @Ignore
  def "Personal top three does not mutate"() {
    given:
    def hs = new HighScores(scores)
    def top3 = hs.personalTopThree()

    expect:
    hs.latest() == expected

    where:
    scores           || expected
    [40, 20, 10, 30] || 30
  }

}
```