JPacMan3 Homework by Tianyu Qi

1. what is the role of EmptySprite?

   1). marking the end of a non-looping sprite

   `AnimatedSprite` class creates a new animated sprite that will change frames every interval. A list of frames is used. When the current frame goes beyond the range of the `animationFrames`, it returns an EmptySprite to mark the end of loop.

   From `AnimatedSprite.java`:

```
private Sprite currentSprite() {
    Sprite result = END_OF_LOOP;
    if (current < animationFrames.length) {
        result = animationFrames[current];
    }
    assert result != null;
    return result;
}
```

   2). the return value when no sprite is created

   `ImageSprite` class creates a new sprite from an image. When splitting a portion of the sprite as a new sprite, the x and y start coordinate and the width and height of the target sprite are needed. If the sprite is not within the image range, return an EmptySprite.

   From `ImageSprite.java`:

```
@Override
public Sprite split(int x, int y, int width, int height) {
    if (withinImage(x, y) && withinImage(x + width - 1, y + height -
1)) {
        BufferedImage newImage = newImage(width, height);
        newImage.createGraphics().drawImage(image, 0, 0, width, height,
x,
                                            y, x + width, y + height,
null);
        return new ImageSprite(newImage);
    }
    return new EmptySprite();
}
```

   From comments of `Sprite.java` interface:

```
/**
 * Returns a portion of this sprite as a new Sprite.
 *
 * @param x
 *            The x start coordinate.
 * @param y
 *            The y start coordinate.
 * @param width
 *            The width of the target sprite.
 * @param height
```

```
     *            The height of the target sprite.
     * @return A new sprite of width x height, or a new {@link EmptySprite}
    if
     *         the region was not in the current sprite.
     */
    Sprite split(int x, int y, int width, int height);
```

2. what is the role of MOVE_INTERVAL and INTERVAL_VARIATION?
    - MOVE_INTERVAL is the base move interval of the ghost.
    - INTERVAL_VARIATION is the random variation added to the MOVE_INTERVAL. This makes the ghosts look more dynamic and less predictable.

    From `Ghost.java` abstract class:

```
/**
 * The base move interval of the ghost.
 */
private final int moveInterval;

/**
 * The random variation added to the {@link #moveInterval}.
 */
private final int intervalVariation;
```

    From `Blinky.java` comments: (Different ghost has different MOVE_INTERVAL and INTERVAL_VARIATION)

```
/**
 * The variation in intervals, this makes the ghosts look more dynamic
and
 * less predictable.
 */
private static final int INTERVAL_VARIATION = 50;
```

3. if you wanted to add a fruit, which files would you need to change?

    Adding fruit is like adding a new kind of pellet. So we can search for the key word *pellet* and create similar methods.

    - create `Fruit` class extending Unit abstract class to give image, value to the fruit.
    - In `LevelFactory`, create `createFruit()` method, and set `FRUIT_VALUE`.
    - In `PlayerCollisions` class, create `fruitColliding(Fruit, Unit)` method to deal with collision, and create `playerVersusFruit(Player, Fruit)` method to show the actual case of player consuming fruit.
    - In `PacManSprites` class, create `getFruitSprite()` method to get the sprite for the fruit.
    - In `Level` class, create `remainingFruit()` class to count the fruit remaining on the board.