1. What is the role of EmptySprite?

`EmptySprite.java` was created by implementing an abstract interface `Sprite.java` and overrides its functions to output nothing (both values of width and height is 0). It is used in 2 places: split function in `ImageSprite.java` and assigned to the *END_OF_LOOP* variable in `AnimatedSprite.java`.

1. In the split function shown below, the main function is to update the sprite appearence by changing x and y coordinates of a image source file. A new EmptySprite will be returned if x and y coordiates are NOT within the image (out of boundary), meaning nothing will be drawn of that sprite on the game panel. For example when the sprite dies, it will be erased after iterating through its dying process in the source image.

```
@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();
}
```

2. In the 2nd class the *END_OF_LOOP* is assigned with an EmptySprite as a static empty sprite serving as the end of a non-looping sprite. The fucntion currentSprite returns the current frame of the sprite. Initially an empyt sprite was assigned to the returned variable and in the situation when the current frame reaches the end of whole frame length, an empty sprite will be return, meaning the sprite disappear from the panel.

```
private static final Sprite END_OF_LOOP = new EmptySprite();
```

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

2. What is the role of MOVE_INTERVAL and INTERVAL_VARIATION?

MOVE_INTERVAL:

It defines the base movement interval, meaning how fast the sprite moves. Different ghosts have their own movement interval and it is fixed

INTERVAL_VARIATION:

While each ghost has its own base movement interval, the another part of their actual moving pattern is defined by a random number between 0 and its own INTERVAL_VARIATION. The following function defines the actual time delayed between moves in milliseconds.

```
public long getInterval() {
  return this.moveInterval + new Random().nextInt(this.intervalVariation);
}
```

3. If you want to add a fruit, which file do you need to change?

1. Add reference to the source image in `PacManSprites.java` by creating an `addCherrySprite` function.

   ```
   public Sprite getCherrySprite(){ return loadSprite("/sprite/cherry.png"); }
   ```

2. Since a fruit is similar to a pellet where it has points and pellet value (10) is initialized in `LevelFactory.java`, a static varible should be created to define how many scores can pacman get by eating a cherry, for example, 50 points. Then similar to the function that creates a pellet, funtion that creates a cherry needs to be created with the cherry value.

   ```
   private static final int CHERRY_VALUE = 50;
   ```

   ```
   public Pellet createCherry() {
     return new Pellet(CHERRY_VALUE, sprites.getCherrySprite());
   }
   ```

3. Next, add cherry on the map when creating the board. Adjust the `board.txt` and change some pellets to "C" (cherry).
4. Add interpretations of symbol "C" in `MapParser.java`.

```java
case 'C':
    Square cherrySquare = boardCreator.createGround();
    grid[x][y] = cherrySquare;
    levelCreator.createCherry().occupy(cherrySquare);
    break;
```

5. Cherry is shown on the board and worth 50 points.