10 Object-oriented Programming: SVG Logo Maker

Your Task

Your task is to build a Node.js command-line application that takes in user input to generate a logo and save it as an SVG file. The application prompts the user to select a color and shape, provide text for the logo, and save the generated SVG to a `.svg` file.

Because this application won't be deployed, you'll need to provide a link to a walkthrough video that demonstrates its functionality and passes all of the tests. You'll need to submit a link to the video **and** add it to the README of your project.

Refer to the Video Submission Guide on the Full-Stack Blog for additional guidance on creating a video.

> **Note**: There is no starter code for this assignment.

User Story

```
AS a freelance web developer
I WANT to generate a simple logo for my projects
SO THAT I don't have to pay a graphic designer
```

Acceptance Criteria

```
GIVEN a command-line application that accepts user input
WHEN I am prompted for text
THEN I can enter up to three characters
WHEN I am prompted for the text color
THEN I can enter a color keyword (OR a hexadecimal number)
WHEN I am prompted for a shape
THEN I am presented with a list of shapes to choose from: circle, triangle, and square
WHEN I am prompted for the shape's color
THEN I can enter a color keyword (OR a hexadecimal number)
WHEN I have entered input for all the prompts
THEN an SVG file is created named `logo.svg`
AND the output text "Generated logo.svg" is printed in the command line
WHEN I open the `logo.svg` file in a browser
THEN I am shown a 300x200 pixel image that matches the criteria I entered
```

Mock-Up

The following image shows a mock-up of the generated SVG given the following input entered by the user: `SVG` for the text, `white` for the text color, `circle` from the list of shapes, and `green` for the shape color. Note that this is just an image of the output SVG and not the SVG file itself:

Additional Requirements

This Challenge combines many of the skills covered so far. In addition to the User Story and Acceptance Criteria, we've provided some guidelines to help you get started.

Because this Challenge requires a video submission, refer to the Full-Stack Blog video submission guide for guidance on creating and sharing a video.

Your application should use Jest for running the unit tests and Inquirer for collecting input from the user. The application will be invoked by using the following command:

```
node index.js
```

It is recommended that you start with a directory structure that looks like the following example:

```
.
├── examples/          // Example svg file(s) created with the app
├── lib/               // Folder for classes or functions
    ├── shapes.js      // Exports `Triangle`, `Circle`, and `Square` classes
    ├── shapes.test.js // Jest tests for shapes
    └── more...        // Additional files and tests
├── .gitignore         // Indicates which folders and files Git should ignore
├── index.js           // Runs the application using imports from lib/
├── package.json
└── README.md          // App description, link to video, setup and usage instructions
```

> **Important**: Make sure that you remove `dist` from the `.gitignore` file so that Git will track this folder and include it when you push up to your application's repository.

> The application must include `Triangle`, `Circle`, and `Square` classes, as well as tests for each of these classes using Jest. While not a requirement, it is recommended that you place any common functionality and properties shared by the `Triangle`, `Circle`, and `Square` classes in a parent `Shape` class and use inheritance to reuse the code in the child classes.

Each shape class should be tested for a `render()` method that returns a string for the corresponding SVG file with the given shape color.

The following example test should pass:

```
const shape = new Triangle();
shape.setColor("blue");
expect(shape.render()).toEqual('<polygon points="150, 18 244, 182 56, 182" fill="blue" />');
```

You may need to add additional files in the `lib` folder for handling user input, writing to a file, etc. Writing tests for these additional files is **optional**.

Helpful SVG Resources

- Example SVG
- Scalable Vector Graphics (SVG)
- SVG tutorial
- Basic SVG shapes
- Text in SVG
- SVG VS Code extension

Grading Requirements

> **Note**: If a Challenge assignment submission is marked as “0”, it is considered incomplete and will not count towards your graduation requirements. Examples of incomplete submissions include the following:
>
> - A repository that has no code
> - A repository that includes a unique name but nothing else
> - A repository that includes only a README file but nothing else
> - A repository that only includes starter code
>
> This Challenge is graded based on the following criteria:

Deliverables: 15%

- At least one sample SVG file generated using the application must be submitted.
- Your GitHub repository containing your application code.

Walkthrough Video: 32%

- A walkthrough video that demonstrates the functionality of the SVG logo maker and passing tests must be submitted.
- The `README.md` file must include a link to the walkthrough video.
- The walkthrough video must show all tests passing from the command line.
- The walkthrough video must demonstrate how a user would invoke the application from the command line.
- The walkthrough video must demonstrate how a user would enter responses to all of the prompts in the application.

- The walkthrough video must demonstrate a generated SVG file, showing the file being opened in the browser. The image in the browser must reflect the choices made by the user (text, shape, and colors).

Technical Acceptance Criteria: 40%

- Satisfies all of the preceding acceptance criteria plus the following:
  - Uses the Inquirer package.
  - Uses the Jest package for a suite of unit tests.
  - The application must have `Triangle`, `Square`, and `Circle` classes.

Repository Quality: 13%

- Repository has a unique name.
- Repository follows best practices for file structure and naming conventions.
- Repository follows best practices for class/id naming conventions, indentation, quality comments, etc.
- Repository contains multiple descriptive commit messages.
- Repository contains a high-quality readme with description and a link to a walkthrough video.

Review

You are required to submit the following for review:

- A walkthrough video that demonstrates the functionality of the application and passing tests.
- At least one sample SVG file generated using your application.
- The URL of the GitHub repository, with a unique name and a README describing the project.