COMP258 Object Oriented Programming Assignment

Value — **Due Date**

10% — 9am, Monday 9 June 2014

This is an individual assignment for Alajmi,Munahi, ID 1076506.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value 10%

Due Date 9am, Monday 9 June 2014

This is an individual assignment for Alattas,Hamid, ID 1168436.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — 10%

Due Date — 9am, Monday 9 June 2014

This is an individual assignment for ALJELAUD,MOHANNAD, ID 1172204.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism
As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule
Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission
Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for Almansour,Abdulmajeed, ID 1060728.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — 10%

Due Date — 9am, Monday 9 June 2014

This is an individual assignment for Almogel,Abdalmohsen Nasser, ID 1155650.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — **Due Date**

10% — 9am, Monday 9 June 2014

This is an individual assignment for Alnazha,Ali, ID 1063738.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value	**Due Date**
10%	9am, Monday 9 June 2014

This is an individual assignment for Burgess,Aimee, ID 1207257.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — **Due Date**

10% — 9am, Monday 9 June 2014

This is an individual assignment for Cameron,Luke, ID 1163194.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for Chai,Yaohui, ID 1180975.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value
10%

Due Date
9am, Monday 9 June 2014

This is an individual assignment for Chen,Albert, ID 1086149.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — 10%

Due Date — 9am, Monday 9 June 2014

This is an individual assignment for Cooksley,Jack, ID 1205824.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — **Due Date**

10% — 9am, Monday 9 June 2014

This is an individual assignment for Dillon,Martin Lee, ID 1126678.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value	**Due Date**
10%	9am, Monday 9 June 2014

This is an individual assignment for Donovan,Jamie, ID 1208700.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value	**Due Date**
10%	9am, Monday 9 June 2014

This is an individual assignment for Eveleens,Greg, ID 1177316.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions
You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app
You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class
I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value **Due Date**
10% 9am, Monday 9 June 2014

This is an individual assignment for Ferguson,Jesse, ID 1204727.

Aim
Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps
The following steps are a likely good order in which to do things.

1. Make up your questions
You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app
You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class
I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value
10%

Due Date
9am, Monday 9 June 2014

This is an individual assignment for Frederiksen,Bjarne, ID 1187730.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for Gilliver,Craig, ID 1162603.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value	**Due Date**
10%	9am, Monday 9 June 2014

This is an individual assignment for Herbes,Josh, ID 1206342.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — **Due Date**

10% — 9am, Monday 9 June 2014

This is an individual assignment for Hope,Troy, ID 1159158.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value
10%

Due Date
9am, Monday 9 June 2014

This is an individual assignment for Kennedy,Finn, ID 1205898.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for Li,Yizhou, ID 1167588.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value	**Due Date**
10%	9am, Monday 9 June 2014

This is an individual assignment for Longden,Mark, ID 1167760.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value	**Due Date**
10%	9am, Monday 9 June 2014

This is an individual assignment for Maguire,Paul, ID 1059261.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value

10%

Due Date

9am, Monday 9 June 2014

This is an individual assignment for Marsh,Rebekka, ID 1162353.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value
10%

Due Date
9am, Monday 9 June 2014

This is an individual assignment for Mayne,Nathanael, ID 1212622.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for Monks,Hamish, ID 1131607.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions
You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app
You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class
I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**

10% | 9am, Monday 9 June 2014

This is an individual assignment for Monteith,Lily, ID 1210367.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value
10%

Due Date
9am, Monday 9 June 2014

This is an individual assignment for Neal,Martin, ID 1182742.

Aim
Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps
The following steps are a likely good order in which to do things.

1. Make up your questions
You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app
You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class
I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value
10%

Due Date
9am, Monday 9 June 2014

This is an individual assignment for Norman,Jared, ID 1189325.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**

10% | 9am, Monday 9 June 2014

This is an individual assignment for O'Connor,Ryan, ID 1210898.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for Old,Jeff, ID 1180804.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for Raju,Sid, ID 1136012.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions
You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app
You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class
I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — **Due Date**

10% — 9am, Monday 9 June 2014

This is an individual assignment for Salter,Richard, ID 1245367.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value

10%

Due Date

9am, Monday 9 June 2014

This is an individual assignment for Sinclair,Kelly, ID 1211769.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for Siwanzi,Henry, ID 1154578.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value
10%

Due Date
9am, Monday 9 June 2014

This is an individual assignment for Steel,Georgia, ID 1190971.

Aim
Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps
The following steps are a likely good order in which to do things.

1. Make up your questions
You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app
You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class
I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**

10% | 9am, Monday 9 June 2014

This is an individual assignment for Stephens,Pearce, ID 1192872.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value	**Due Date**
10%	9am, Monday 9 June 2014

This is an individual assignment for Wang,Wenbo, ID 1154318.

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value
10%

Due Date
9am, Monday 9 June 2014

This is an individual assignment for Wu,Gaby, ID 1164350.

Aim
Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps
The following steps are a likely good order in which to do things.

1. Make up your questions
You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app
You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class
I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value **Due Date**
10% 9am, Monday 9 June 2014

This is an individual assignment for Yu,Shaohui, ID 1216584.

Aim
Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps
The following steps are a likely good order in which to do things.

1. Make up your questions
You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app
You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class
I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — **Due Date**

10% — 9am, Monday 9 June 2014

This is an individual assignment for „ ID .

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for ,, ID .

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for „ ID .

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — 10%

Due Date — 9am, Monday 9 June 2014

This is an individual assignment for „ ID .

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value **Due Date**

10% 9am, Monday 9 June 2014

This is an individual assignment for „ ID .

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value **Due Date**

10% 9am, Monday 9 June 2014

This is an individual assignment for „ ID .

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value | **Due Date**
10% | 9am, Monday 9 June 2014

This is an individual assignment for ,, ID .

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link:
http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value — **Due Date**

10% — 9am, Monday 9 June 2014

This is an individual assignment for „ ID .

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value
10%

Due Date
9am, Monday 9 June 2014

This is an individual assignment for „ ID .

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions
You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app
You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class
I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.

COMP258 Object Oriented Programming Assignment

Value	**Due Date**
10%	9am, Monday 9 June 2014

This is an individual assignment for „ ID .

Aim

Create a Processing application for a Road Code Quiz.

The Road Code Quiz should have 10 questions. The user is prompted firstly to start the quiz, and then to answer each question one at a time. After a response is given for each question, the user is told that they either (i) answered the question correctly or (ii) answered incorrectly, in which case they are shown the correct answer.

At the end of the quiz the user is shown a summary of how well they did in a graphical way using a gauge.

For an example of a quiz with the same specifications, take a look at this link: http://www.aa.co.nz/drivers/driving-school/road-code-quiz/

Steps

The following steps are a likely good order in which to do things.

1. Make up your questions

You should make up your own questions based on the NZ road code, which can be found online here: http://www.nzta.govt.nz/resources/roadcode/

2. Obtain some images to enhance your app

You are permitted to use images from the road code link above (use of which will *not* be considered plagiarism and they don't need to be referenced).

You are also permitted to use any images for graphics and icons (e.g. ticks and crosses) obtained via Google image search with usage rights set to "Labeled for reuse".

3. Design a Button class

I have shown you an example of a Button class in lectures, but do not use that example. Instead, design your own Button. Note that my example lacks many sophisticated features such as responding automatically to mouse rollovers and having images as backgrounds. See the "Start quiz" button on the example quiz above.

4. Design a Gauge class
A Guage object is used to display the user's final score out of ten at the end of the quiz. A minimal Guage would be a circle, a needle, and some labels positioned around the guage so that the final score can be easily read.

6. Design a Question class
A multi-choice question consists of (i) text, (ii) an image, (iii) four possible answers, (iv) a specification of which answer is correct, and (v) a state (either answer shown or answer hidden).

7. Design a Quiz class
A Quiz is a collection of Questions. It should have properties for both the user's current score (number of correct responses) and also know which part of the quiz (i.e. introduction page or question number or final page) is currently being displayed. In other words the Quiz class should be a container for the Questions and other objects.

8. Put it all together
Once the classes are designed and tested, you can put them all together into a finished app. Make sure you test the app thoroughly before submission!

Use of Inheritance/Polymorphism

As it stands, the above program can be constructed without inheritance and polymorphism. However, for full marks, use **inheritance** to make your quiz alternate between two different classes of question: (i) multiple choice questions where the user chooses one of four answers, and (ii) short answer questions where each question is a single word answer typed in by the user. The short answer question will require a new control, namely a text box. Your program should have an abstract superclass for both classes of question.

Marking Schedule

Marks for this assignment will be assigned in the following way:

- Steps 1-8 – 60%
- Extension using inheritance to multiple question classes – 15%
- Good design of classes, properties and methods; and quality of code (naming, comments etc) – 15%
- Design, appearance and professionalism of the UI – 10%

Note that you **must use object oriented programming (classes, objects, methods, inheritance etc) to solve this problem**. You could write a quiz program without classes (i.e. COMP103 style) but you will be penalized significantly if you do that.

Submission

Submit a zipped version of your Processing project to the moodle handin box by the due date.