University of BRISTOL

Software Engineering Project Report: Mulan

Team Hummingbird

Hui Yan, Jinlu Li, Yue Wang, Defa Qian, Pingyang Zhao, Han li

15/05/2018

Declaration of Marks

We declare that the following distribution of marks have been agreed by all of our team members.

HUI YAN	**100%**
YUE WANG	**100%**
JINLU LI	**100%**
HAN LI	**100%**
PINGYANG ZHAO	**100%**
DEFA QIAN	**100%**

Acknowledgements

Thanks to Dr Ian Holyer who provides us advices and guidance for the project as well as the knowledge of Java.

Contents

1. Introduction

1.1. Backgrounds Information

In this software engineering course, the aim is to produce a game or application which is attractive to youngsters, but also has an educational element, exploring some aspect of computer science "how can computing concepts be demystified to a young audience?"

Today's computer games are becoming far more complicated and are played regularly by millions of people of all age levels around the world, especially youngsters. But for our concern, some games are meaningless for students except fun. Therefore, we think it would be better to add some value to the game so that it won't be a waste of time. A game that contains some cool computer science knowledge is what we are going to build.

Moreover, in September 2013 the UK Department for Education published a document: Computing programmes of study: key stages 3 and 4, saying that" A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world.[web ref1]"

And the document indicates that the national curriculum for computing aims to ensure that all pupils:

- can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
- can analyse the problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
- can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
- are responsible, competent, confident and creative users of information and communication technology.

Generally, it is significant to build a game that helps the students learn the computer science knowledge and inspire their interest in programming. Therefore, this game

we are going to build has two foundations: computer science related and educational oriented function.

1.1.1. Computer Science Related

In this software engineering group project, we are tending to develop a game that related to computer science knowledge. But computer science has a very wide range of scope and we must focus on a specific aspect which should not be obscure to youngsters.

1.1.2. Educational Oriented Game

This game that we are going to develop should contain the educational function that helps to teach young children to learn computer science knowledge. This game is not just for fun. It has meaning that can attribute to the children future development with programming skill.

Value-added research shows that the most promising features of games use conversational language, put words in spoken form, add prompts to explain, add advice or explanations, and add relevant pregame activities. Cognitive consequences research shows that first-person shooter games improve perceptual attention skills. Media comparison research shows that games are more effective than conventional media for science learning [1].

In this game project, we shall decide which kind of game we are going to develop, which computer science knowledge we are going to introduce to the audience and what effects we are going to impact on the audience.

After several discussions, we decide to develop a game that similar with the Scratch which teaches children how to write a program. And the main knowledge relevant to computer science is the idea of object-oriented programming. We hope this game would introduce this key fact of programming clearly and funny.

1.2. Aim and Objectives

From the unit scope, the aims are to: develop an understanding of software development gain experience with teamwork, techniques, tools become better prepared for development in industry.

The sort of things we should understand are:

- advanced programming issues
- object-oriented design
- software lifecycles
- agile development

From the project scope, the aims are: design and develop a complete game with chapters of stories, more specifically, are:

- Game structure
- Game interface
- Game story and Characters
- Game interaction
- Testing and evaluation

Generally, in this software engineering project, we as a team are going to implement our program skills to build an educational game that related to object-oriented programming knowledge. Hopefully, this project would not only benefit our software development skill but also add value to the computer science education.

2. Project Initialisation

2.1. Project Overview

After several meetings and the first presentation, we almost settle down what we tend to do in this project. We divide this project into 5 parts: analysis, design, development, testing and deployment.

Analysis: In this initial stage, first we evaluate the potential audience and the market. Then we have a brainstorm of what we tend to build and decide the final idea and concept of the game. After that, we write the game outline and structure to build the prototype lately.

Design: We investigate the idea of the Scratch game and its background by searching some articles. Then we figure out which game parts we need to implement our project. After deciding what to do next, we organize the code structure and find out the libraries needed to use. We also allocate the programming into pairs.

Development: This is the main part of the project. We start writing codes according to what we have designed. We do the pair programming and debugging together in a study room in queen's building. We also have Discuss & Integrate

Testing: We are going to implement the Transparent tests, Opaque tests, Unit testing and the Beta testing.

Deployment: Package the program into Jar file. And prepare for the final report.

2.2. Development Language

In this project, we are going to use JavaFX for development. JavaFX is a software platform for creating and delivering desktop applications, as well as rich Internet applications (RIAs) that can run across a wide variety of devices. JavaFX is intended to replace Swing as the standard GUI library for Java SE, but both will be included for the foreseeable future [2]. Therefore, we use JavaFX instead of Swing for this project.

Some JavaFX advantages:

- Solid API with bundled inbuilt features for almost anything you will need.
- Brilliant Graphics Rendering (2D).
- Excellent memory management.

FXML is an XML-based user interface markup language created by Oracle Corporation for defining the user interface of a JavaFX application. It provides a convenient alternative to constructing such graphs in procedural code and is ideally suited to defining the user interface of a JavaFX application since the hierarchical structure of an XML document closely parallels the structure of the JavaFX scene graph. However, anything that is created or implemented in FXML can be expressed using JavaFX directly [3].

FXML Cons: It takes slightly longer to load and display.

FXML Pros:

- Rapid scene development / mock up using Scene Builder.
- FXML is not a compiled language; you do not need to recompile the code to see the changes. Just reload the FXML file.
- It provides a clear separation of GUI from logic/controller.
- You can have different versions of a scene/view using the same controller. This is handy for demos for instance.
- The content of an FXML file can be localized as the file is read.

2.3. Development Environment and Tools

We are going to use the IntelliJ IDEA IDE which is convenient, and we are familiar with to build our game. We have already updated the Java 8 library, so everything is ready.

We also use OneDrive for backup, Git for version control, Wechat for communication and Google Docs for report writing.

2.3.1 OneDrive:

We chose OneDrive as a tool for us to share our working documents. The main use of it was allocating works to every teamers and checking tasks. In additions, after every meeting, we uploaded the records of meetings on time.

Conveniently, everyone can modify the working documents so that can improve the efficiency and avoid some troubles.

This software is developed by Microsoft. There are some useful functions for our group members shown below:

1. The automatic backup of album, that is, without manual intervention, OneDrive automatically uploads pictures on the device to the cloud for storage, so that even if the device fails, teamers can still obtain and view pictures from the cloud.

2. With the online Office function, Microsoft combines Office software with OneDrive. Users can create, edit, and share documents online. WE can also perform arbitrary switching with local document editing, save them locally, or save them online. The online edited file is saved in real time, which can avoid the loss of the file content caused by the downtime during local editing and improve the file security.

3. To share a specified file, photo, or entire folder, we can just provide an access link for sharing content to other teamers. Other teamers can only access these shared content, and cannot access non-shared content.

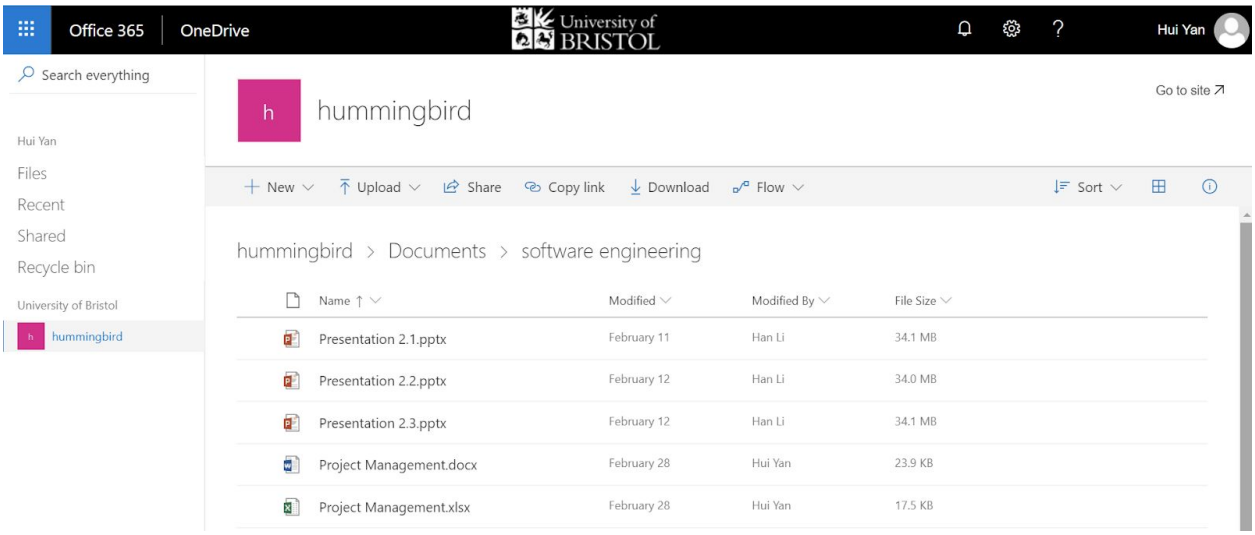


Figure 1: OneDrive

2.3.2 Git and Github:

When working with a group on the same project, the version control is considerably essential. In this way, all group members can finish every part of the program nearly at the same time, we do not need to wait for others to finish earlier part. As for the tool of version control, we decided to use Git(on github.com).

In this way, it could guarantee that our workload are small when reviewing code, which could also help improve the quality of code and reduce the conflict. In addition, it could help us have a better understanding of respective work areas, then when we may simultaneously modify the same file, we would generally be aware of other teamers what reforms will be more effective, for example: if there is something wrong with merge, how should we solve? Moreover, it could also unify everyone's coding format, so that could avoid a lot of meaningless conflicts. Finally, it was important that we need to keep a log of working versions from the past to go back on when somethings went wrong.

To have a better understanding of Git, we all studied online to get grips with the git commands, Git is seen as an industry standard of version control, we all agree this experience using it in a group project can help us a lot in the later work. As shown below.

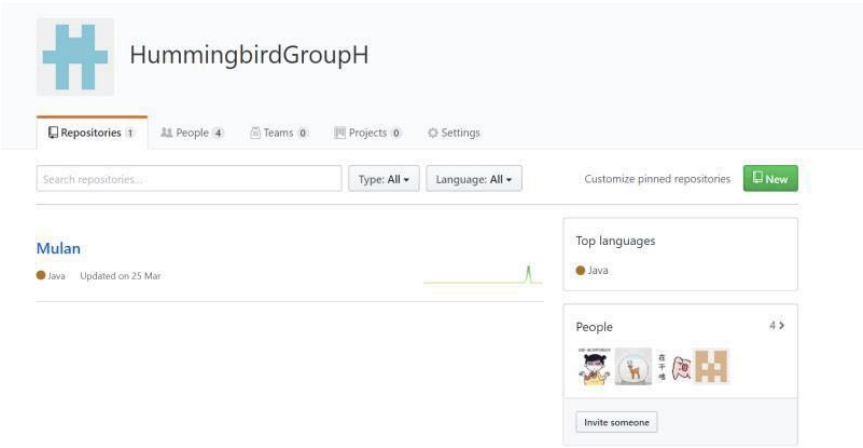


Figure 2: Github

2.3.3 WeChat

As for the tool of communication, we chose WeChat, which is an application of Chinese multi-purpose SMS, social media and mobile payment developed by Tencent. It is convenient for us to discuss with each others at anytime so that can accelerate the progress of this project. Moreover, when someone did not stay in Bristol, we could also conduct a video conferencing which can benefit us a lot. In addition, we can also share some small work documents on it just by mobile phone sometimes.

2.3.4 Google Docs

Google Docs is a web-based word processing and spreadsheet program that can considerably improve collaboration efficiency. As for why we chose this tool, this project belongs to a team work, we want to put our documents online. In this way, other people in the team can edit and update the document online in real time, and it can also eliminate the need for us to send the same version of the document offline and update the same document repeatedly based on each different reply. Multiple team members can change files online at the same time, and can see the edits made by other members in real time. The system automatically saves each revision so that we can see who made the changes and when they can revert to the old version at any previous point in time. As shown below.

Figure 3: Google Docs

3. Concept Designing

3.1. Game Objectives

As mentioned above, this game shall introduce the object-oriented programming knowledge to the youngsters. Therefore, this is the main objective of this game. Besides this, the game should be fun enough so that it can attract the audience and make them keep playing. Nobody loves a boring game.

3.2. Target Client and Market Analysis

This game obviously targets the students in school. However, as the object-oriented programming is an abstract concept that requires some basic computer science knowledge to understand, the target audience should not be too young, therefore, high school students are the appropriate audience and this game may raise their interest in computer science.

3.3. Game Concept

At first, we search for some popular games that teach children about programming knowledge. There are two games that impressively attract our eyes, the Tynker and the Scratch, which are both teaching children to code interestingly.

Tynker is a complete learning system that enables everyone from beginners to advanced programmers to code with intuitive, interest-based activities. Tynker's self-paced courses let kids experiment with visual blocks before progressing to intermediate programming and even languages like JavaScript and Python [web ref2].

Figure 4: Tynker

Scratch is a visual programming language and online community targeted primarily at children. Using Scratch, users create their own interactive stories, games and animations, then share and discuss their creations with one another. Developed by the Lifelong Kindergarten group at the MIT Media Lab, the service is designed to help children (ages 8 and up) learn to think creatively, reason systematically and work collaboratively [web ref3].

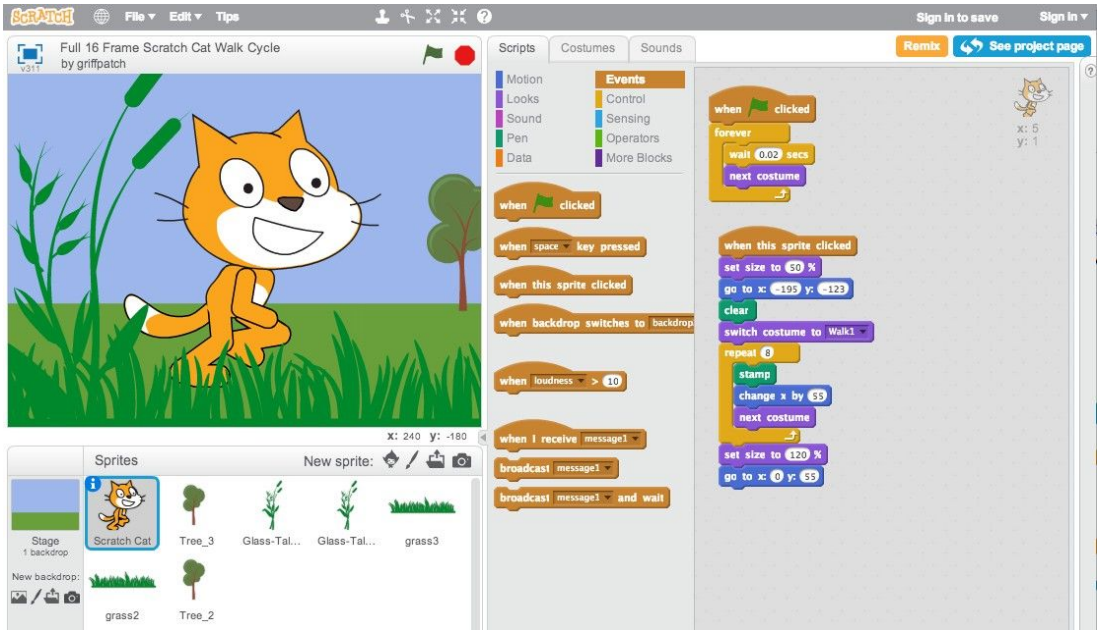


Figure 5:Scratch

After investigating these two games, we gather together and have a meeting. In the meeting, we have several interesting game idea after the brainstorm:

- Computer science quiz competition for children: Interesting at the beginning but requires enough knowledge
- Get out of the maze: hard to relate to computer science education
- Typing game: hard to relate to computer science education as well
- Sorting game: sort what?
- Computer decryption game: decrypt what?
- Computer structure game (like a LEGO): not attractive enough with several levels
- Chinese programming game (Help Chinese primary students who don't understand English(terminology) learning basic computer skills): A good idea but we want to do something

We draw all the idea on a whiteboard, list out each SWOT analysis results, and then debate which one is the best. However, we cannot agree on each other, so we pick out three idea, vote for two to present on the fist presentation. After the presentation, it seems that the Scratch like game based on the Mulan story is the most welcome idea. Therefore, finally we decide that we are going to develop a game similar to the Scratch that teaches object-oriented programming skill.

Figure 6:Example of Scratch

Scratch is a programming language that is perfect for making games, animations, interactive stories and other visually rich programs[4]. It provides a great introduction to programming for people of all ages. It's widely used in schools and colleges, but Harvard University has also used it in higher education at its Summer School. There are workshops for adults where Scratch provided a friendly introduction to the kind of creative problem solving that programmers do all the time.

Scratch is easier to use than most other programming languages for a number of reasons:

• You don't have to remember or type any commands: they're all on screen, so you can just drag and drop them.

• Commands fit together like jigsaw pieces, so there are strong visual hints about how you can combine them.

• Error messages are rare. Because Scratch commands lock together, programs always make some kind of sense. It is possible to still write programs with logical errors in, if they don't do what you expected, but Scratch guides you to write things that work, rather than nagging you when they don't.

• The commands are colour-coded and categorized, so you can easily find a command when you need it.

• The commands in Scratch simplify common activities in games, such as testing whether a missile has hit an alien (collision detection), or rotating a character on screen.

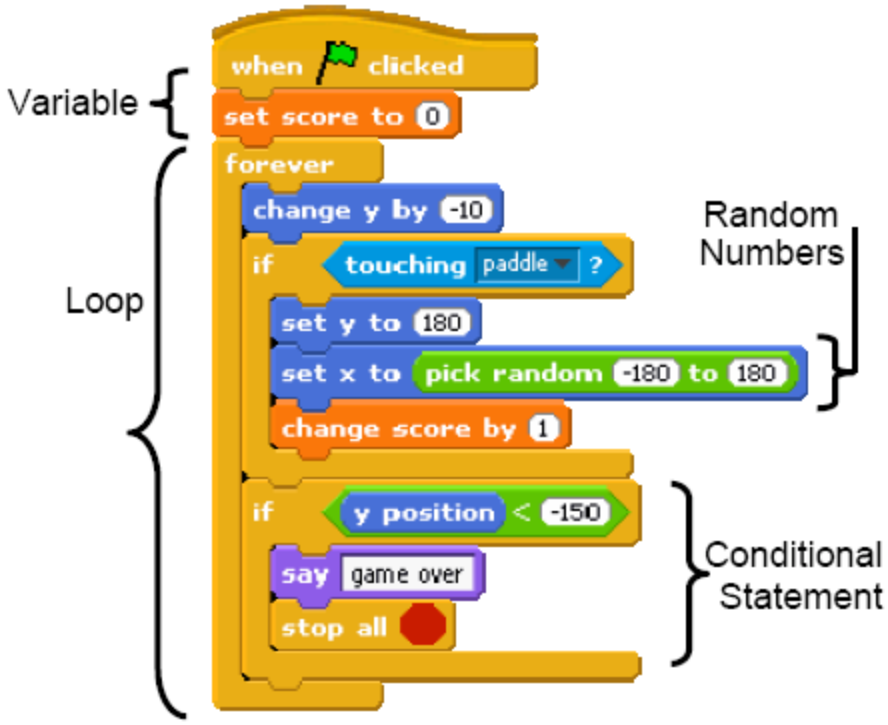


Figure 7: Example of Scratch

In short, Scratch is designed for programming learner. It enables the user to quickly see results from the work, and even includes graphics and sounds. Many other programming languages require learners to learn text commands and strict rules about how to use them. Scratch doesn't. That means the audience can focus their energy instead on the fun stuff: dreaming up ideas for new programs, working out how to build them, designing them, and sharing them with friends.

3.4. Game Type and Style

After analyzing current programming-teaching software such as Scratch, we decided to develop a 2D drag and drop game with Mulan's story, which is easier for users to understand the Object Oriented Programming language features. And this game will introduce the Java programming language by guiding users to answer each chapter's drag questions during a story.

After we decided to develop a game that have some educational meanings, the first priority of this project is how to develop a character that can meet our demands and meantime can attract the interests of users. Then after a period of a week, we all

agreed that Mulan Hua is the best choice for us. Because our teamers are all Chinese, we want to add some Chinese elements to this project so that can introduce our culture to the users. In addition, Mulan Hua has always been a woman respected by the Chinese for thousands of years, because she is brave and simple. In this way, she could be seen as a represent of Chinese culture.

The background of this game is a traditional Chinese story about Fa Mulan, shows in the icon. The further information will be introduced in the next section. Mulan's story is well-known in China, and this story has animated musical action comedy-drama film by Disney in 1998. Since this film has excellent music and character design, we decided to use some materials from this film in order to introduce this Mulan story vividly.

We divide the Mulan story into seven chapters, and add some introductions of Java features base on the story. These chapters contain beginner level and advanced level. After accomplishing these chapters, users will have a general understanding of Java concepts include the three main features of Java, inheritance, encapsulation and polymorphism.

3.5. Characters and Story

The story begins with the Huns invade China by breaching the Great Wall in Han Dynasty, the Chinese emperor orders a general mobilization. Conscription notices require one man from each family to join the Chinese army.

When Fa Mulan hears that her elderly father Fa Zhou, the only man in their family as an army veteran. Mulan becomes anxious and apprehensive due to her father's weakening health. So Mulan decide take her father's place. After the training camp, Mulan is able to pass as a man. Mulan must conceal her identity and kill her enemies with her partners which is even more difficult than what the average

soldiers does. As the reinforcements solemnly leave the mountains, Mulan's army are ambushed by the Huns, but Mulan cleverly uses a cannon to cause an avalanche, which buries most of the invaders.

Mulan is forgave of her crime of bullying because of her great merit and praised by the Emperor. The assembled inhabitants of the city bow to her in an unprecedented honor. The Emperor gives the crest and the sword to Mulan as gifts and believes that Mulan has the ability to serve in the court and take an official position. However, Mulan refuses the request because she needs to take care of her family. The emperor allows Mulan to return home so that she can compensate and honor her family. Finally, Mulan returns home and presents the gifts to her father, who is more overjoyed to have Mulan back safely.

This story may be new to most of the non-Chinese students, so it may raise their curiosity about the character and be attracted by the different culture style.

3.6. Interaction

As shown below, users can drag the labels which they need to use in one specific section to the text area, then click ‘done’ button. In this way, users can finish every section of the story clearly.

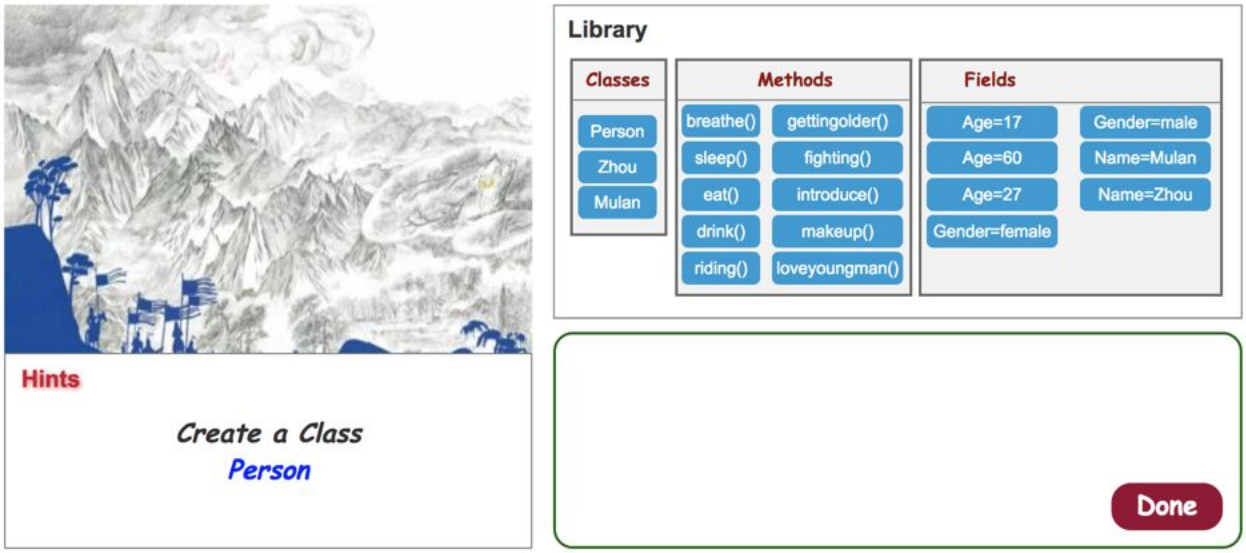


Figure 8:early version game scene

3.7. User Interface

We decided to add a start menu to the game, this menu is composed of the picture of Mulan Hua and 'start' button. This picture of Mulan can attract the interests of users considerably before starting the game. In addition, when clicking 'start' button, users can enter the game directly. It is shown below.

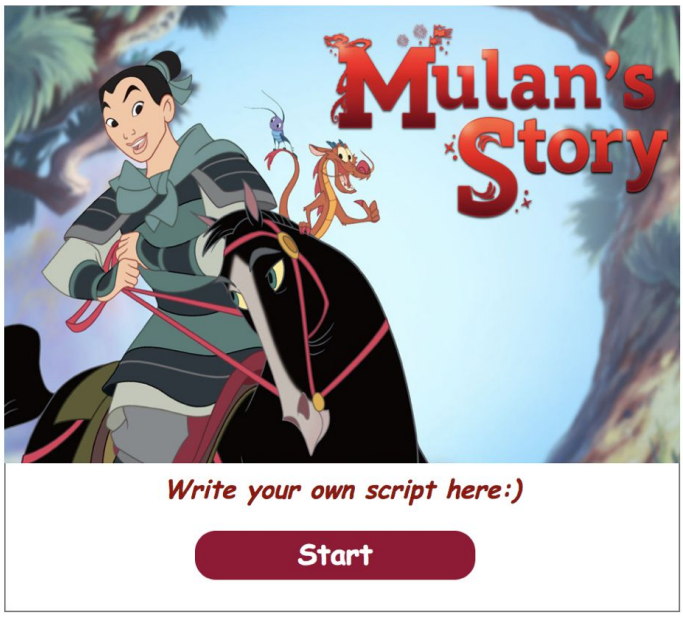


Figure 9:early version start scene

3.8. Conclusion

In conclusion, this Scratch like the game project we are going to do is based on the heroine story of Mulan. By dragging and dropping different methods or functions according to the hints, the player can conduct the character to achieve the goal of the storyline. And the drag and drop process will involve in the object-oriented programming method unconsciously to the player. Therefore, by playing this game, the players, in another word, targeted students, are going to learn the concept of object-oriented programming method without notice.

SWOT Analysis

Strength:	**Weakness:**
• Good storytelling • Very education • Related to Computer Science	• A little bit complicated at the beginning and not straightforward • May wear off quickly after the story ends

• Inspire the creativity of the player to build their own story of the game • Inspire the interest of the player in programming	• Limit the creativity of the player if the hints are too strict
Opportunities: • Introduce the OOP concept to the audience without their conscious • Showing a culture diversity	**Threats:** • Hard to combine the story with the concept of Java • How to make a balance between tutorial and game elements

4. Project Design

Java is a general-purpose programming language developed with the aim to bring portability and a higher level of security. [5] There are three main features of Java which make it become a unique and popular programming language. In order to introduce the Object Oriented programming Java clearly, we decided to explain the Java features through the Mulan's story in seven chapters. This game will mainly introduce the three features of Java: Inheritance, Encapsulation and Polymorphism.

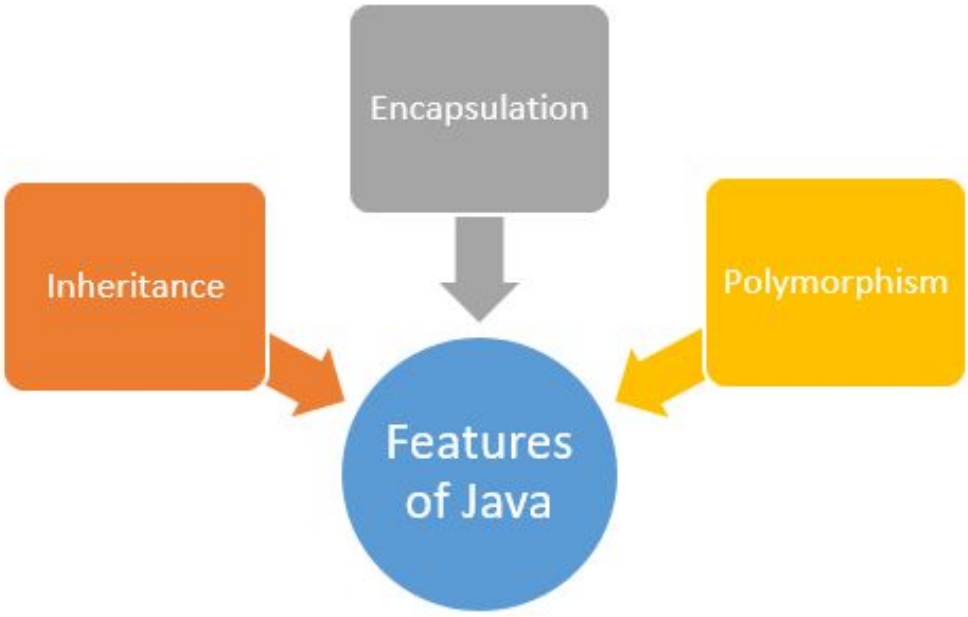


Figure 10: Three main features of Java.

Since the audience users of this game are high school teenagers, we decided to illustrate these Java technologies with hints and explain tutorials. In chapter one, as for the basic Java concepts. Since an object can be anything and object have states and behaviors. In Java object states called Fields and object behaviors called Methods, indicate the things which object has and what object can do. Java uses Class which is a blueprint for an object. We utilise the example of a normal "Person", a Person can breathe, sleep, eat and drink etc. Each of these things can be a method in the "Person" class.

Chapter 1: Person

Storyline: Mulan's story begins.

Knowledge point: OOP programming.

Hints: give a picture with a person on it., saying "I am a person. Guess what things I can do?"

Question Setting: Choose one class and choose methods for it.

Correct answer: Person is the correct answer for "Class" type. Some verbs for human such as: "eat", "drink", "breathe", "sleep" are correct answers for "Method" class.

The confusing options:

Class:

Some specific names such as "Mulan", "Zhou" would teach users to distinguish information from hints pictures and make preparation following story plot.

Some verbs and adjectives to help users learn that the "Class" corresponds to the "object".

Method:

Some verbs such as "bark", "soar" to teach users know that the methods are what the "object" can do.

Some noun and objectives to help users distinguish that methods are behaviors.

Explain & Tutorial: OOP concept, how to create the class, how to add methods and field in it.

Discussion point:

The question setting: whether add field in the answer area?

Adding field will make this chapter complete in introducing the basic element in the Java; However, there seems no suitable fields for "Person" class. Maybe "name", "age" etc. are feasible but the field in Java is more likely to be "identifier = value", which would also cause ambiguity in the following chapter.

Final decision: do not add fields in chapter one but give some simple explanation and introduction about fields in the tutorial part.

Chapter 2: Zhou

Storyline: Introduce Zhou. Zhou is the father of Mulan and is brave and loyal to the emperor, but he is too old to fight.

Knowledge point: Inheritance.

Hints: give a picture with Zhou on it., saying “I am Zhou. I am 60 years old. I am an army veteran. Help me finish my ‘methods’ and ‘fields’.”.

Question Setting: Choose one class and choose methods and fields for it. The class is extended from Person and it should be provided, for example, ”Person”, “Extended” are already in the answer sheet or in the hints picture.

Correct answer: “Zhou” is the correct answer for “Class” type. Some verbs that meet his veteran status such as “fight”, “ride a horse”, “hold a sword” are correct answers for “Method” class. Fields: name=”Zhou”, age=60 etc.

The confusing options:

“Class” setting is similar to the chapter one.

“Method” setting will improve difficulty and are all verbs for human. The verbs that obviously used for other professions such as “cook”, “code” can be added to help users more involved in the story plot.

“Field” setting is similar to “Method”, the form would be strictly same as code in Java.

Explain & Tutorial: Inheritance concept, show the code in Java when creating a class. Some explanation for type for data in Java, such as “String”, ”int” and so on.

Discussion point:

The question setting: how to represent the gender of Zhou?

The “gender=male” would be clear in library area but if we want to show some code in Java related to it would be difficult because make the type of gender to be String would be strange. In fact, the gender in this game would only be female and male. Thus, making the gender data type to be Boolean maybe a good choice.

Final decision: make the gender type to be Boolean and introduce the Boolean type in tutorial part.

Chapter 3: Mulan

Storyline: Mulan is ready to see the match maker. In ancient China, a girl can bring her family honor by striking a good match. Her character has shown on the way she goes to the match maker's house. She is naughty, honest and brave.

Knowledge point: Inheritance (the difference between override methods and new methods).

Hints: give a picture with Mulan on it, saying "I am Mulan. I am 17 years old. Help me finish my 'methods' and 'fields'.".

Question Setting: Choose one class and choose methods and fields for it. The answer sheet area has provided "Zhou extends".

Correct answer:

Class: "Mulan" is the correct answer for "Class" type.

Method: Some methods that "Zhou" has will be needed to teach the concept "override". Some new methods that can meet Mulan's status, such as make up, love young man etc.

Field: the identifiers are same as fields of "Zhou".

The confusing options:

Similar to chapter 2.

Explain & Tutorial: Override the methods when the subclass has the same methods as superclass.

Discussion point:

The question setting: how to teach users to learn modifiers in JAVA such as private, public in this chapter?

If we make all the fields to be private, we will not access it easily in the following chapter. We could use "get()" method, but it seems that it will make our tutorial more complicated.

Final decision: we would not put the modifiers "private" or "public" in the tutorial(Convert to java part). Once we need call methods, we will mention that but we will not focus on explaining modifiers.

Chapter 4: Recite Rules

Storyline: Mulan have to meet the Matchmaker.

Knowledge point: Encapsulation.

Hints: Choose the right method. And Mulan saying “Oops, I have to answer the Matchmaker’s questions. Please help me.”

Question Setting: Choose one method to help Mulan.

Correct answer: Choose the correct Method “reciteRules”.

The confusing options: The most confusing one is “makeCheatSheet()”. The question is related to the story so it is clear that the match marker tells Mulan to recite the rules. The match maker “call” the method “ReciteRules”. However, match maker does not know how Mulan recite the rules. Maybe Mulan studies hard to memorize the rules, maybe she make a cheat sheet. The caller neither cares nor knows the contents of this method. That is also the key point of this chapter.

Explain & Tutorial: Encapsulation concept, how it achieved in Java. For instance, we can call the Method “reciteRules” without know the string of “cheatSheet” in it. This chapter also introduce public Method concept and method requests return value.

Discussion point:

The tutorial: How to introduce the feature of Encapsulation?

Since Encapsulation is used to hide the internal representations. In other words, we can call the method to use, but we cannot know the detail things in this method.

Final decision: let the users to choose the right Method “reciteRules” without know the “cheetSheet” in it and introduction the Encapsulation concept in the tutorial part.

Chapter 5: Mulan joins the army

Storyline: Since the Huns invade the Great Wall, the emperor require one man from each family to join the army. Mulan's father Zhou, who is the only man in their family. But Zhou is too old to fight. So Mulan decided take her father's place.

Knowledge point: Up casting fields in Polymorphism.

Hints: give a picture with Mulan in soldier uniform, saying "I am soldierMulan I will take my father's place. How do I introduce myself (fields)?"

The upcasting sentence should also be given:"Zhou soldierMulan = new Mulan();"

Question Setting: Choose the correct fields of up casting object "soldierMulan".

Correct answer: "soldierMulan.name=" should match "Zhou", "soldierMulan.age=" should match 60 and "soldierMulan.gender=" should match "male".

The confusing options:

This chapter aims to teach upcasting.By casting it is not actually changing the object itself, we are just labeling it differently. We create a "Mulan" as "soldierMulan" and upcast it to "Zhou", then the object doesn't stop from being "Mulan". It's still Mulan, but it's just treated as Zhou and it's properties of Mulan are hidden until it's downcasted to not soldier Mulan again.

This chapter would focus on telling users that Mulan's properties are hidden so Mulan's properties and Zhou's properties should be provided to confuse the users.

Explain & Tutorial: Polymorphism upcasting, the fields in the upcasting Object are same as superclass fields. And introduce a test method which can help users to check whether the hypothesis is correct.

Discussion point:

The tutorial: How to introduce the feature of Polymorphism?

Since Polymorphism is an important feature in Java, which is help Java become more flexible.

Final decision: let the users choose the right field of "soldierMulan", help the users to understand the main features of upcasting can try to understand the concept through the

story of Mulan take her father's place and pretend her father (use the same fields of superclass).

Chapter 6: Mulan fights against enemies.

Storyline: As the reinforcements solemnly leave the mountains, Mulan's army are ambushed by the Huns, but Mulan cleverly uses a cannon to cause an avalanche, which buries most of the invaders.

Knowledge point: Up casting Methods in Polymorphism.

Hints: give a picture with Mulan in fighting, saying "I am soldierMulan. It is me again. I am going to fight Guess what things (Methods) I can do?"

Question Setting: Choose the correct Methods of up casting object "soldierMulan".

Correct answer: Methods "soldierMulan" can called are Zhou's introduce and Mulan's fight and rideHorse. It is worth mentioning that the method in Zhou which is static will still work for the object, so the "introduce" of "Zhou" is the correct answer.

The confusing options:

The zhou's methods are provided.

Explain & Tutorial: Polymorphism upcasting, the Methods in the upcasting Object are same as subclass fields. If the method is static, the method will same as superclass. And introduce the specific Methods in Mulan subclass cannot be used after upcasting.

Discussion point:

The Explain&Tutorial: Is there important to introduce the real Java code or just shows the pseudo-code.

Final decision: Introduce the real Java code which can help users understand the operating principles of Java.

Chapter 7: Mulan comes back home

Storyline: Mulan returns home and present the gifts to her father, and she can act as a girl.

Knowledge point: Down casting of Polymorphism.

Hints: give a picture with Mulan dressed like a girl, saying “I am mulan. I am back after 10 years! I can act as myself. Guess what things (Methods) I can do?”

Question Setting: Choose the correct Fields and Methods of down casting object “mulan”.

Correct answer: “mulan” can use all the functions in “Mulan” subclass.

The confusing options:

Zhou’s methods and fields are provided to confuse the user. It would help the user know the concept of “Downcasting”. When an object downcasts to the right level, it can use its own methods and have its properties.

Explain & Tutorial: Polymorphism downcasting, the Methods and fields in the downcasting Object are same as subclass fields.

5. Project Management and Planning

5.1. Development Strategy(Agile)

The introduction of the concept of "Agile" was firstly introduced from the field of software development[6]. The traditional software development adopts a waterfall-type development process[7], which divides the entire development process into a few phases such as: collecting requirements, definition, design, coding, testing, and releasing. At every stage, developer sets clear goals and standards, and then enters the next stage. In this way, the entire process moves in the direction of increasing predictability, which can avoids inefficient investment of resources and effectively guarantees the quality of development.

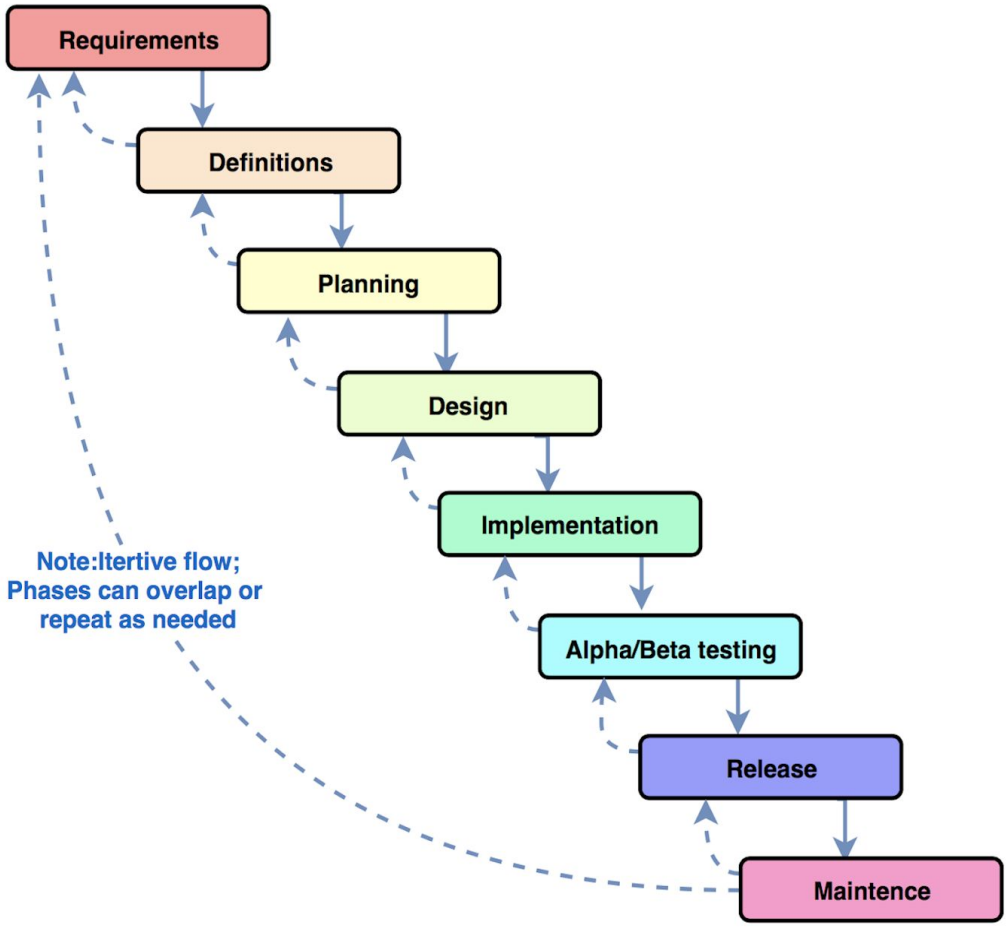


Figure 11: the progress of waterfall-type development.

However, there is a problem that lies in the waterfall-type approach to the development of this pre-defined process. Because there is a strong relationship of dependency between every stage, the earlier stage is considered as the input of the next stage, if the quality of the input is not high, it will seriously affect the quality of output of the follow-up stage. At the same time, if the earlier stage fails to meet the standards, it will cause some stagnations in the subsequent stage, which can lead to a longer development cycle. Moreover, because of the commitments made in the early stages of the project, it is considerably difficult to adjust the changes in demand in the later period.

Under such a circumstance, the Agile-type development was born. Researches have shown that seventy percent of software development projects using waterfall-type development method failed[8]. The reason lies in the fact that the demands of market always change rapidly and it is difficult to achieve a clear and complete collection of product requirements. At the same time, the development of technology is increasingly fast, in this way, there are multiple uncertainties regarding the attainability of the defined functions. Therefore, when the collections of requirements and the definitions of product cannot be completed well, the waterfall-type development method naturally cannot escape high failure rates.

Therefore, from the two dimensions of the definiteness of the demand and the certainty of the realization of the project, when the ambiguity of the demand and the uncertainty of the implementation exceed a certain range, the characteristics of the complex system (the complex system) will be exhibited, then the waterfall-type development will also be no longer practical. In this way, the Agile-type development method was born under this circumstance.

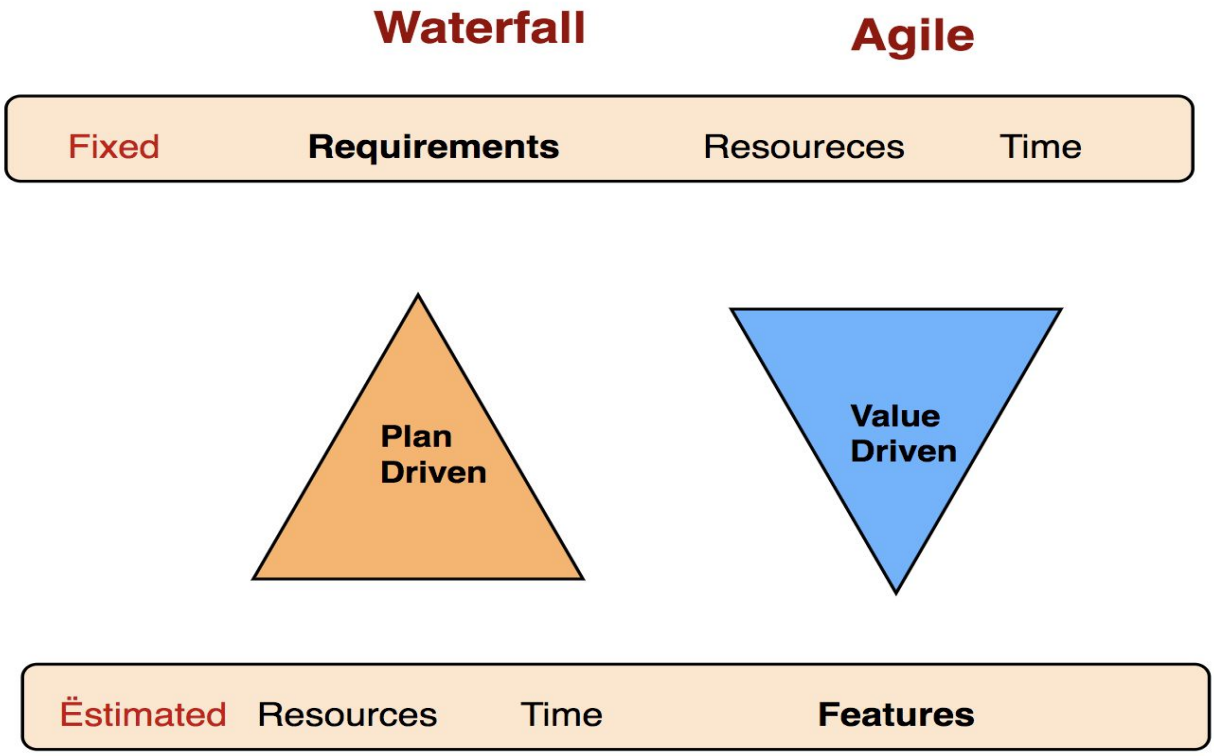


Figure 12: Agile vs. waterfall

The transformation of a core thinking mode of Agile-type development is from "Fix Scope, Flex time" represented by waterfall-type development to "Fix time, Flex Scope" which means fixed time, elastic range. In the context of market changes and technological changes, since the “scope” represented by the demand of market and the definition of product cannot be solidified, it is impossible to determine how much resources should be invested to complete it. In this way, we can try to fix existing resources and limited resources, making the "range" achieve maximum. Therefore, we can shift the project from "plan-driven" to "value-driven."

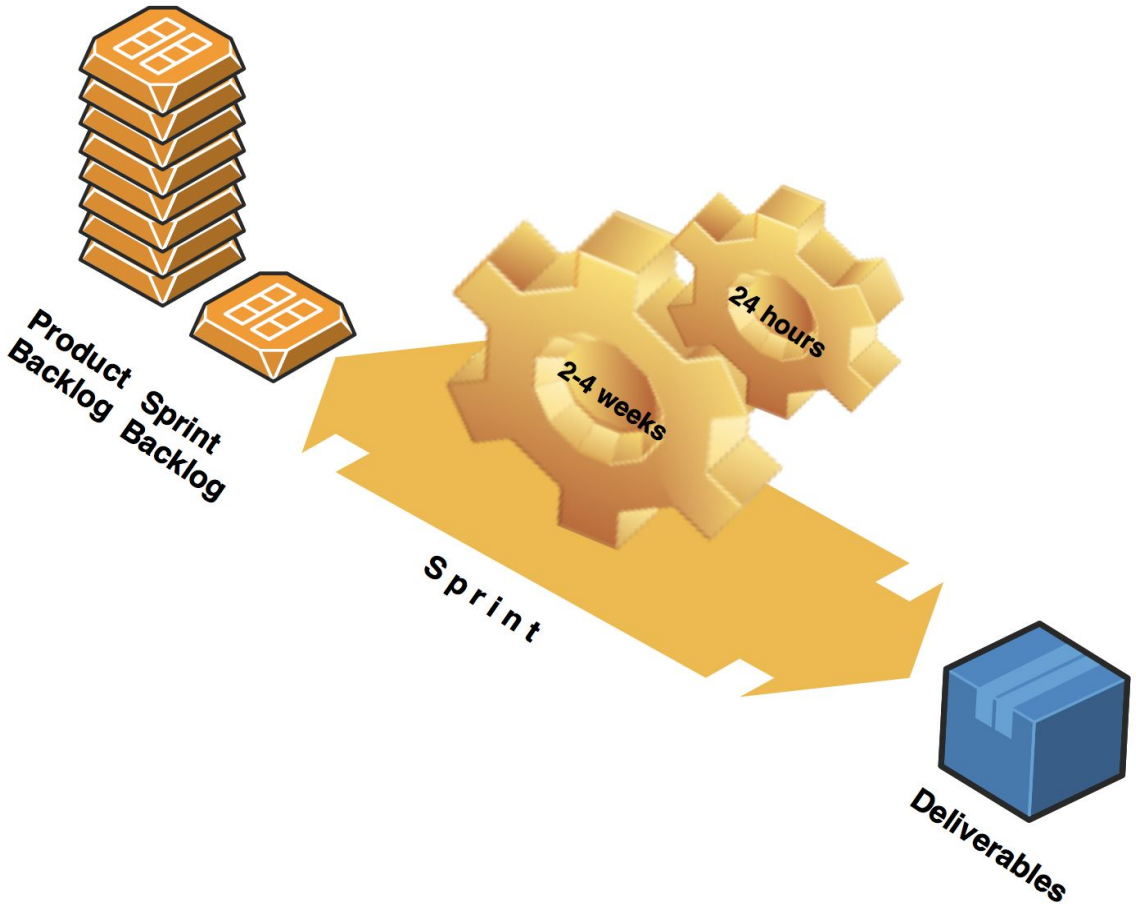


Figure 13: Workflow

Compared to the engineering method of the pre-defined process represented by waterfall-type development, the Agile-type development method is closer to the final environment of the application through test driven/value-driven means, thus this application can have a better adaptability. At the same time, under the guidance of the Agile Manifesto, we can put more emphasis on the value of the code writer and better tap the potential of code writers.

Figure 14: Agile vs. engineering

5.2. Version Control

As for the version control, our group chose git to complete this part. As mentioned in section 2.3, git can help us share our work documents, such as code, back up of the whole progress and save previous versions. In this way, if some troubles happen, we can also guarantee the project go on well.

However, there was a problem that any one of our group did not have any experience of using git, so we took a few days to learn how to use git together, and then had a better understanding of git, which benefited us a lot at the late stage of the project.

5.3. Tasks Allocation

Work Breakdown Structure & Responsibility Matrix

Id	Task	Id	Subtask	Schedule	Responsibility					
					Yue	Hui	Lu	Yang	Han	Fa
1	Analysis	1.1	Brainstorm and decide the final idea	02.05	✓	✓	✓	✓	✓	✓

		1.2	Prepare for the 1st presentation	02.06 –02.11	✓	✓	✓	✓	✓	✓
		1.3	Design the game	02.12 –02.18		✓			✓	
2	Design	2.1	Organize the storyline	02.19–02.25				✓		✓
		2.2	Organize the program structure	02.26–03.04	✓	✓	✓	✓		
		2.3	Allocate programming pairs	03.05	✓	✓	✓	✓	✓	✓
3	Development	3.1	Pair Programming & Debugging	03.06–04.22	✓	✓	✓	✓		
		3.2	Discuss & Integrate	03.06–04.22	✓	✓	✓	✓		
4	Testing	4.1	Unit Testing	04.23-04.30			✓		✓	✓
5	Deployment	5.1	Packaging	04.30	✓				✓	✓
6	Report	6.1	Write the final report	05.01–05.13	✓	✓	✓	✓	✓	✓

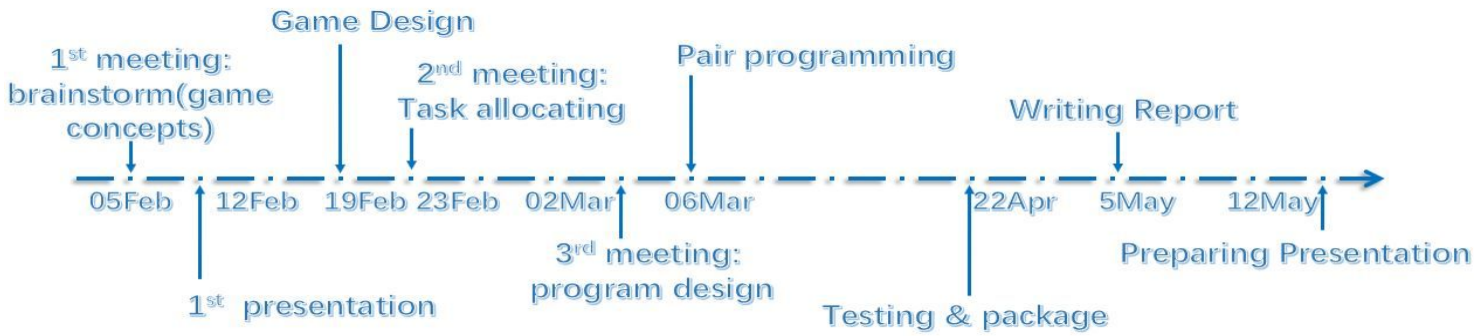


Figure 15: Timeline

6. Project Development

6.1 Overview of Programming Design

To effectively improve program reusability and reduce the program coupling, we decided to use the most well-known pattern, the Model-View-Controller (MVC) pattern, to develop the game[9]. The MVC pattern is an architectural pattern that separates three main groups of components from each other: visuals (Views), data (Models), and logic (Controllers). Each component takes its responsibility to handle a specific aspect of development. Using this pattern, user requests are routed to a Controller which works with the Model to retrieve results of queries or perform user actions. The Controller chooses the View for displaying for the user and provides it with Model data it requires [web ref4]. The advantage of this pattern is that different team members can take responsibilities of different parts of the program in order to make full use of labor and promote efficiency and can be further modified and developed if some changes are applied to one of the concerns in the future.

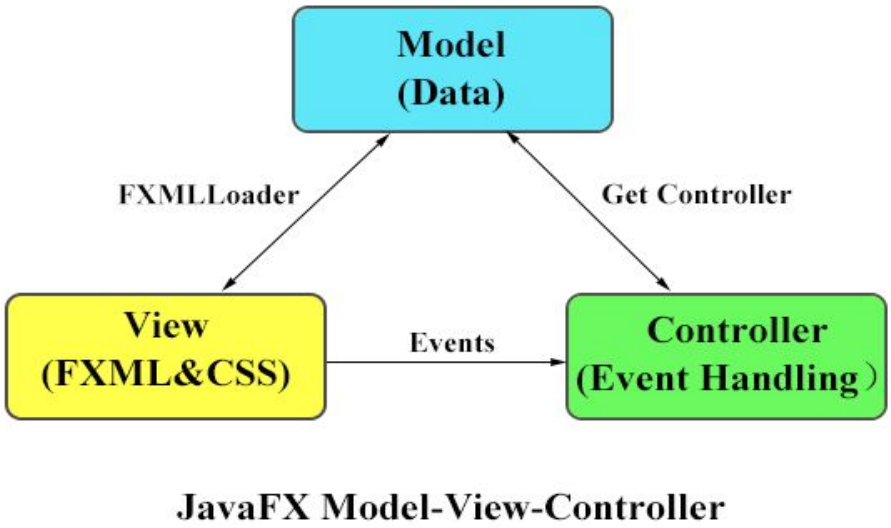


Figure 16: Diagram of interactions within the MVC pattern

The left half of the interfacial structure is image area and the right half is area for user interaction.

The data of the game are divided into three main parts: the string data that are displayed on the labels, the image data that are used as background story, hints and tutorial of the game, and the video data that help push forward the storyline.

The logic of the game is quite simple. Game levels are upgraded along with story by judging whether user's drag-and-drop actions are correct. Some explanations are embedded to help users answer questions and understand knowledge points.

We experienced three versions of programming for polishing our program.

6.2 Version I

Based on the MVC pattern, we placed our file into four folders, which are Controller, CSS, FXML and pics used for storing source files. In addition, we need a Main class to trigger the start of the game.

6.2.1 Data(Models)

In the first version of our development, we found the disadvantages of Controllers during the process of development before we constructed the data structure.

6.2.2 Visuals(View)

In this version, we had not added any visual effects to our game yet. We just placed some empty fxml files in the appropriate package for future development.

6.2.3 logic (Controllers)

As is shown in the following figure, we designed a single controller, a single stylesheet and a single fxml file for each chapter. We made the three parts in a one-to-one mapping relationship.

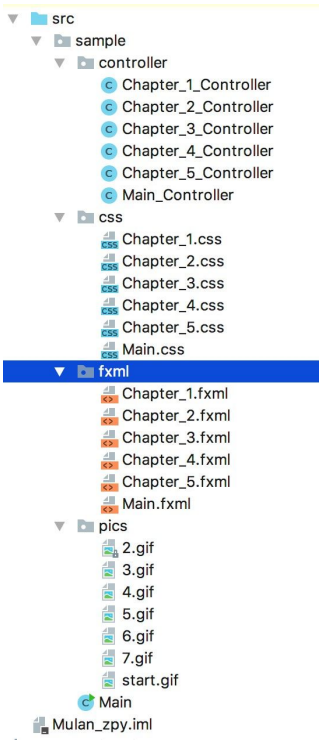


Figure 17: Structure of program in our first version of development

6.3 Version II

6.3.1 Data(Models)

1. words

In this version, we thought about our “words” data on the labels as they are flexible and they are required to change in different levels of the game. In order to keep the program clean and versatile, we organized the data in a four-level hierarchy.

The “words” data in the game has three parts: words that are displayed on the labels in the “Library” panel, “Answer Sheet” panel, and the correct answers that are used to compare with users’ answers.

We have thought about two ways to store those words for game:

(1) Create a class in java which can be used to store those words. Those words can be put in the fields of this class as “String”.

(2) Import those words from a text file. Those words can be stored independently.

At last we chose the second way to implement. It would make our program more robust. Those words can also be modified quickly and conveniently.

In order to make those words organize well and be easy to be called, we learned from the “Db” assignment and extend our work. The structure for the words data is in Figure No.x. The game has seven chapters. Each chapter has three groups: library, answer and key answer. Every group has three types of words: class, method, field. Every chapter’s structure is solid and the contents of them will be shown in the “view” part of the game.

The “Record” class is a list of strings which the size of it is uncertain. The GroupData has 3 Records, and the ChapterData has 3 GroupData. The function related to “words” data is “read”, “write” and “check”. Chapter Data is called by the chapter number, and the Group data and Records are called by the keywords.

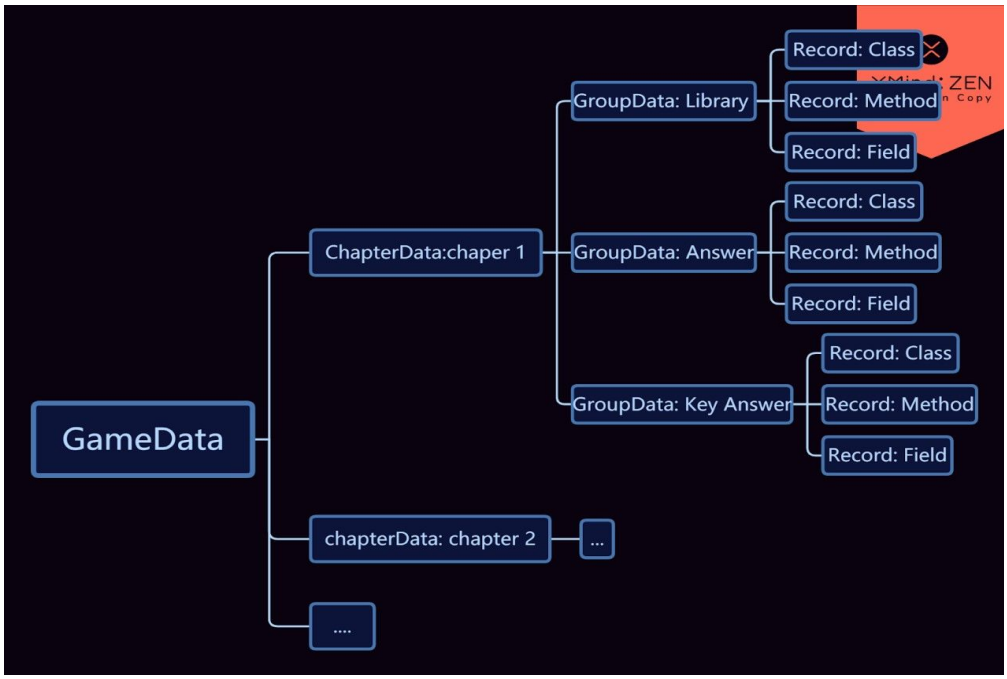


Figure 18: The structure of our “words” data of our game

The way to “read” those words is to put them on the labels of the game. Once the one of the chapters of the game is finished, the “words” will give its value to the labels. We wrote a method “setText” in GameData class to assign words to Texts elements.

The way to “write” the “word” is to read a text file. The labels for the “library”, “answer” and “key answer” are in different files. It would help the contents in three group would not affect each other. At first we have thought about put the library labels and right answers in the same file, for example, put the correct answers as top five words in the “Class” type, then mess up the order to make them show on the game panel. Then we found it would not be clear when we wanted to modify the labels in the “library” and “Answer sheet”.

However, we found that it is not good enough when we design the questions and answers in the game. We had to modify three files when we want to change the questions. The fact is that we could know what would be put into the “Answer sheet” and what is the correct answer when we designed a chapter, so it made a little messy for “writing” in the words data. Then we have arranged to write a method that can separate each chapter by the symbol ”#”. Each chapter recognizes its group by the key words ”Class”, ”Method” and “Field”. It would make it clear when the words are in the file. Every word for label is separated by an exclamation mark. We have put unit testing in this part to make sure that it works well.

The most important part of the game is to check whether the user’s answer is right or wrong. We decided to put this function in the logic part.

2. Pictures

Our game uses many pictures (in PNG and GIF formats) to beautify the game panel. The pictures have been divided into two parts. The hints pictures on the game panel to help users answer the questions. The explanation pictures for the answers of the questions to help users learn more about JAVA skills.

We encountered the difficulty that how to load the path of the pictures in a convenient and reliable way. Then we solve the problem by using a for loop to get the pictures according to the way they are named. To achieve this, we

stored the image data in the order of the chapter number they belonged to. It is effective but fragile, while we did not think of a better way, which was a pity.

Figure 19: Images stored in order of name

```
for(int k = 1;k < 5; k++){
    remoteUrl1.add("file:src/Mulan GIF/00" + k + ".gif");
}
```

Figure 20: For loop for getting the pictures

3. Videos

The videos are imported in the same way as the pictures do. And the video data were stored and loaded in the same way as the pictures did.

6.3.2 Visuals(View)

1. Layout

The rough layout of the main body of the game is made up of four parts. The left-top part is for glancing the idea of the current storyline by placing dynamic graphs, and the left-bottom part is for prompting players what they are required to do by placing pictures. The right-top part acts as a library for accommodating the elements, in text style, which are ready for players to drag. Finally, the right-bottom part acts as an answer sheet which is the destination for dragging elements from the library to.

We used the JavaFXSceneBuilder, which can export file in fxml format, to help sketch the layout. We chose it because it was intuitive and convenient. In the first version, we used the GridPane to divide the scene, VBox to accommodate a single column of texts and HBox to accommodate groups of VBoxes. We used VBoxes and HBoxes because they are good choices for positioning. We assigned an fx: id to each widget for injecting into (by using

notation @FXML) and being used in the program in the future development. The rough glancing of our layout is as shown in the following figure.

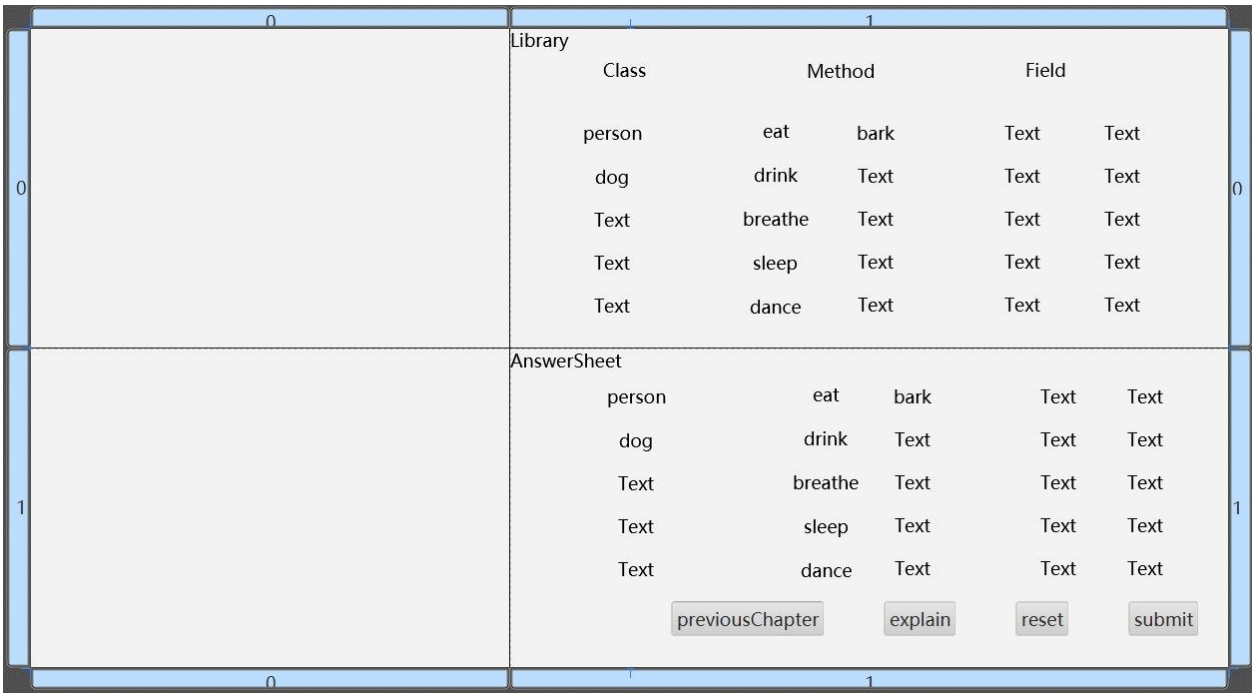


Figure 21: View of fxml file in JavaFXSceneBuilder

2. Add Cascading Style Sheets (CSS)

We added some style effects to the interface by writing a css file and getting style sheet by using the following statement:

```
scene.getStylesheets().add(getClass().getResource( name: "game.css").toExternalForm());
```

Figure 22: The statement for getting stylesheet in the program

The glancing of the interface is shown in the following figure:

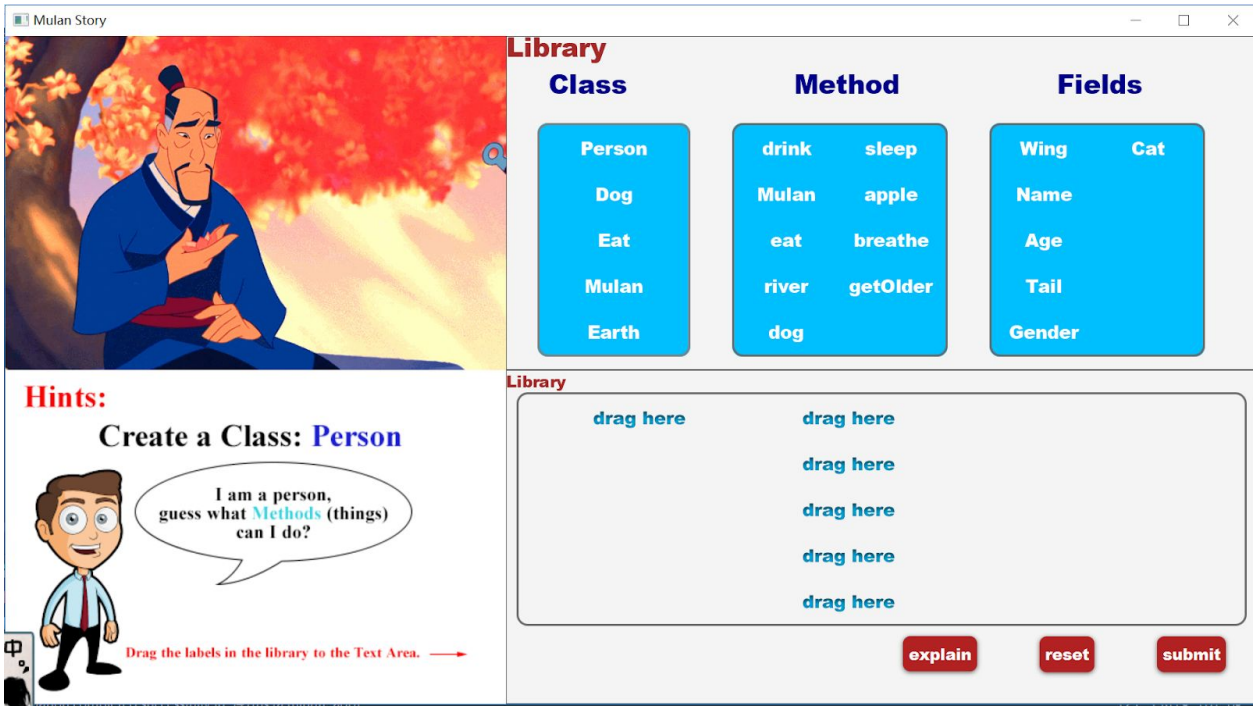


Figure 23: The interface with adding some simple css effects to

In this version, the UI design was roughly according to the prototype. In addition, we added some hover effects to labels and buttons in order to improve user experience by obviously giving users signals that it can be operated. However, restricted to the layout disadvantages, we could not add separate background color to each label, because the labels were just Texts distributed in VBoxes and HBoxes and Texts did not have the background color css property. Also, the interface looked a little bit tedious. In order to make our UI more attractive, we improved our layout and cascading style sheet in our third version of development.

6.3.3 logic (Controllers)

1. Refactoring

We refactored the first version of program to a cleaner one in the process of development. We converted to using only one Controller to control the whole logic. The game had three threads including upgrading game levels, displaying explanations in new pop windows and playing videos for pushing forward the storyline in new pop windows.

2. Core Design

(1) Game upgrading logic

We abandoned switching stages when moving on to the next Chapter. The process of the game levels upgrading was designed to carry on in one scene in one stage all the time, with switching Texts and Images on it only. A variable named “chapterNum” was used to control reading and updating string data and image data when switching chapters. It was also used to control the “previousChapter” button, deciding whether current scene could go back to the previous Chapter and whether the “previousChapter” button was visible (because the first Chapter should not have this button). This improvement needed a reusable underlying data structure to support. This verified the necessity of constructing the underlying data structure.

The core method of the Controller is the “moveOn (ActionEvent event)”. It was used decide whether the game level could upgrade and whether the story could carry on. According to the product design logic, when the “submit” button is pressed, a method is used to collect user’s answers by fx: ids and the check method is utilized to check whether the answers are correct. If the answers are correct, the story video will pop up and the texts on the game scene will change at the same time. Otherwise, an alert will be given to tell players that the answers were wrong and prompt them to utilize the embedded explanations.

```
private void moveOn(ActionEvent event) {
    if (answerChecker.check(answerChecker.collectAnswer(ansCla, ansMet, ansFie), gameData.getKeyAnswer(chapterNum))) {
        videoStage.popVideo(chapterNum);
        ++chapterNum;
        updateText(chapterNum);
        updateImage(chapterNum);
        updateBtn();
    } else {
        alert.showAndWait();
    }
}
```

Figure 24: Check the answers and decide whether move on

(2) Optimize Loading Speed

We had many dynamic pictures and videos to load and they occupied much space. At first, we put the pictures and videos outside the source package in a directory and once the picture needs to be shown, the program will search for it and call it. This could really slow down the execution of the program. Thus, we decided to load the path of the pictures and videos and initialize the elements on the panel before the start of the game. We created two new classes called “SetImage” and “MakeMedia” to help preload pictures and videos in ArrayLists before the main body of game being loaded. In this way, we only need to wait for a while before the game starts and can show images and media instantly during the process of the game.

```
private SetImage setima = new SetImage();
private ArrayList<Image> gifImage = setima.changeImage();
private ArrayList<Image> explainImage = setima.changeExplaImage();
private ArrayList<Image> hintImage = setima.changeHintImage();
private MakeMedia makeMedia = new MakeMedia();
```

Figure 25: Preload images and media

(3) Utilize JavaFX in Good Manner

The core JavaFX facility we used was Drag-and-Drop. As the reason that we had many labels organized in ArrayList form in the library area and answer sheet area, we used “(event -> {})” to achieve the facility so that we could pass parameters into the methods in order to use for-loop to apply the action event to every element in a clearer way, although using “(event -> {})” might not be a quite elegant and recommended way to add events.

```
public void doDraggedDropped(ArrayList<Text> target) {
    for (int i = 0; i < target.size(); i++) {
        int finalI = i;
        target.get(i).setOnDragDropped(event -> {
            Dragboard db = event.getDragboard();
            boolean ic = false;
            if (db.hasString()) {
                target.get(finalI).setText(db.getString());
                ic = true;
            }
            event.setDropCompleted(ic);
            event.consume();
        });
    }
}
```

Figure 26: A method used for applying Drag-and-Drop facility to Texts

For other widgets, we used the recommended style to add events to. For example, the action event for “submit” button was written in the following style. It tells the “submit” button what will happen when it is pressed: switching the chapter by changing texts and images on the scene and popping up a new window for playing the video for pushing forward the story at the same time.

```
nextBnt.setOnAction(this::nextExplain);
previousBnt.setOnAction(this::previousExplain);
```

Figure No.x Pass the callback method to widget

```
private void nextExplain(ActionEvent event){
    iv.setImage(explainImage.get(2*chapterNum-1));
}

private void previousExplain(ActionEvent event){
    iv.setImage(explainImage.get(2*chapterNum-2));
}
```

Figure 27: Callback function: tell what events will happen when this method is passed

3. Difficult Points

For loading one fxml file in main controller, we could use the following statement:

```
Parent root = FXMLLoader.load(getClass().getResource( name: "/game.fxml"));
```

Figure 28: Load fxml file

```
@FXML
private VBox lcb1, lcb2, lcb3, lcb4, lcb5;
@FXML
private VBox lmb1, lmb2, lmb3, lmb4, lmb5, lmb6, lmb7, lmb8, lmb9, lmb10;
@FXML
private VBox lfb1, lfb2, lfb3, lfb4, lfb5, lfb6, lfb7, lfb8, lfb9, lfb10;
```

Figure 29: Declarations for injected values with fx: ids

After injecting the components in a class, we can freely access them in the code.

Because we wanted new windows popping up for displaying explanations and playing videos separately and they both have their own fxml file, we had to load more than one fxml files in one main Controller java class. However, the previous loading statement and notation @FXML did not work when we attempted to load two fxml files into one controller. This confused us for some time before we eventually found the solution. To load the fxml file, we should use the following statement:

```
FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource( name: "/explainpopwindow.fxml"));
GridPane cmdPane = fxmlLoader.load();
```

Figure 30: Load the second fxml file

At this moment, components in the fxml file could not be injected into the code by using notation @FXML any longer. We could not use fx: ids directly, either. We had to create a new object and use “lookup” to find the component by fx: id instead.

```
Button nextBnt = (Button) popScene.lookup( selector: "#next");
Button previousBnt = (Button) popScene.lookup( selector: "#previous");
```

Figure 31: Lookup widget in the fxml file by using fx: id

6.4 Version III

6.4.1 Data(Models)

In this version, the data of the game are divided into three main parts: words, pictures and video. The words or phrases are used to be shown on the labels and the image data that are used as background story, hints and tutorial of the game, and the video data that help push forward the storyline.

Figure 32: The whole data structure in our game, including words, images and videos

1. Words

We added a new class “TextSetter” as a tool (put in the “tool” package) to assign the words to the Text elements. We replaced the method “setText” from the GameData Class to the new class. In this way, our classes could be more concentrated which meant that a particular class did a particular thing.

2. Pictures

We gave the “SetImage” class a new name called “ImageData” and put it into the “data” package. Although this is just a simple movement, it meant a practice of improving the structure and make packages better organized.

3. Videos

The video data were relocated in the same way as the pictures did.

6.4.2 Visuals(View)

1. Layout

(1) Reason for Abandoning JavaFXSceneBuilder

After a discussion between our group members, we found that although JavaFXSceneBuilder was a competent tool for designing UI, it would generate a certain number of useless tags and unnecessary attributes automatically such as <children> tag and node orientation properties. This could make the code in a messy manner. Therefore, we decided to abandon using this tool and turn to coding fxml files directly. The two figures below represent the difference between the two ways directly. The fxml file coded by us is much neater than that generated by JavaFXSceneBuilder.

```
<VBox GridPane.rowIndex="1" GridPane.columnIndex="0"  id="right_c1" GridPane.columnSpan="2">
    <Text fx:id="lc" id="title" text="CLASS"></Text>
</VBox>
<VBox GridPane.rowIndex="2" GridPane.columnIndex="0" fx:id="lcb1" id="ct1">
    <Text fx:id="c1" text="person" id="ct"></Text>
</VBox>
<VBox GridPane.rowIndex="3" GridPane.columnIndex="0" fx:id="lcb2" id="ct1">
    <Text fx:id="c2" text="dog" id="ct"></Text>
</VBox>
<VBox GridPane.rowIndex="4" GridPane.columnIndex="0" fx:id="lcb3" id="ct1">
    <Text fx:id="c3" text="xxx" id="ct"></Text>
</VBox>
<VBox GridPane.rowIndex="5" GridPane.columnIndex="0" fx:id="lcb4" id="ct1">
    <Text fx:id="c4" text="xxx" id="ct"></Text>
</VBox>
<VBox GridPane.rowIndex="6" GridPane.columnIndex="0" fx:id="lcb5" id="ct1">
    <Text fx:id="c5" text="xxx" id="ct"></Text>
</VBox>
```

Figure 33: The fxml file written by hand

```
                                    <Text fx:id="tf9" strokeType="OUTSIDE" strokeWidth="0.0" text="Text">
                                       <VBox.margin>
                                          <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
                                       </VBox.margin>
                                    </Text>
                                    <Text fx:id="tf10" strokeType="OUTSIDE" strokeWidth="0.0" text="Text">
                                       <VBox.margin>
                                          <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
                                       </VBox.margin>
                                    </Text>
                                 </children>
                              </VBox>
                           </children>
                           <HBox.margin>
                              <Insets bottom="20.0" left="20.0" right="20.0" top="20.0" />
                           </HBox.margin>
                        </HBox>
                     </children>
                  </HBox>
                  <HBox alignment="TOP_RIGHT" prefHeight="100.0" prefWidth="200.0">
                     <children>
                        <Button fx:id="explain" mnemonicParsing="false" text="explain">
                           <HBox.margin>
                              <Insets bottom="10.0" left="10.0" right="10.0" />
                           </HBox.margin>
                        </Button>
                        <Button fx:id="reset" mnemonicParsing="false" text="reset">
                           <HBox.margin>
                              <Insets bottom="10.0" left="10.0" right="10.0" />
                           </HBox.margin>
                        </Button>
                        <Button fx:id="submit" mnemonicParsing="false" text="submit">
                           <HBox.margin>
                              <Insets bottom="10.0" left="10.0" right="10.0" />
                           </HBox.margin>
                        </Button>
                     </children>
                  </HBox>
               </children>
            </VBox>
         </children>
      </GridPane>
```

Figure 34: The fxml file generated by JavaFXSceneBuilder

In our implementation, we separated the css and layout from each other strictly. No style statements were written in the fxml files.

(2) Design

In this version, we improved our layout in order to make it more versatile. We put each Text label in its own VBox so that it had a container that could be added some styles to. And the right-top and right-bottom parts of the interface concluded a new small GridPane respectively. This made the positioning more flexible because our Texts on the labels were not in the same length.

2. Improve Cascading Style Sheets (CSS)

(1) Label Background

After we improved the fxml file, our labels could be set background colors and background radius to make it in a round-corner rectangle shape. This made the labels more like labels and could improve user experience.

(2) No-Border Design

We found that many popular games do not have borders in their window mode. To make it more like a game instead of a coursework, we removed the border of some windows.

(3) 3D Effect

JavaFX is not a proficient engine for game development like Unity3D or Cocos-x. To make players immerse themselves in the plot of our game, we designed animated background. We made the background show the same dynamic picture as in the left-top area and set it transparent to make it look like the shadow of the picture. In this way, we had tried to imitate 3D effects to make the UI more attractive.

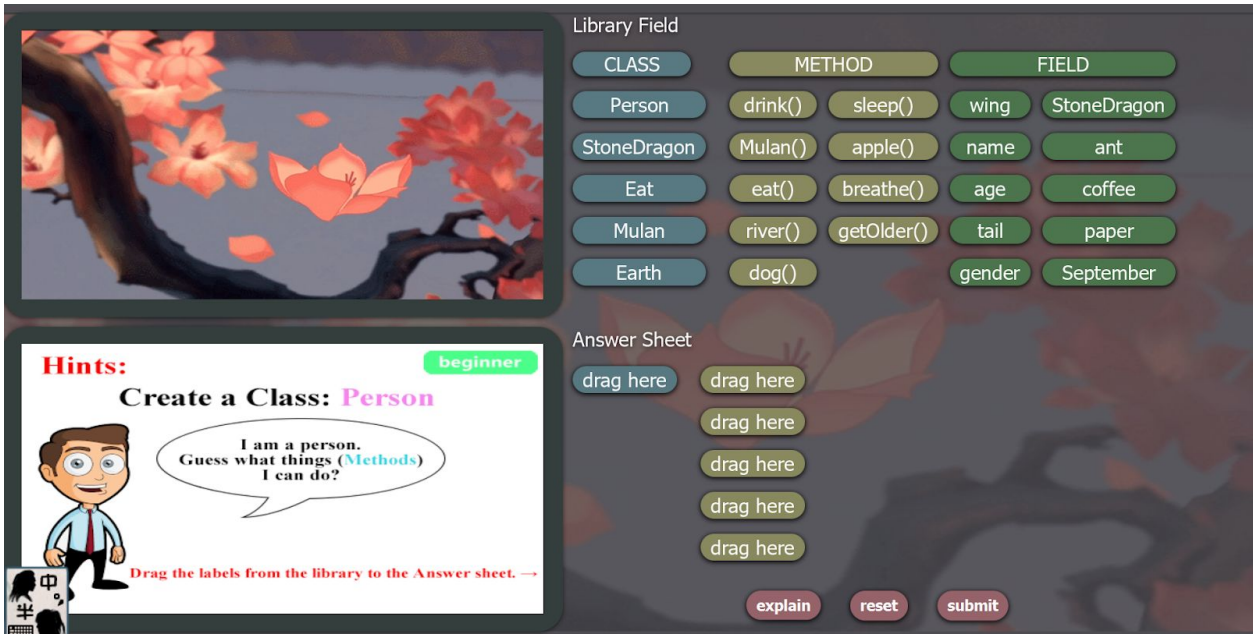


Figure 35: The final interface of our game body

(4) Visual Upgrading

We applied shadow effects to buttons to make it look like floating on the screen visually. In addition, borders were added to pictures to make the area more like media player.

6.4.3 logic (Controllers)

In this version, we again refactored our program. We made the specific package contain the specific classes for doing the specific things. The package “assets” was for storing the data including “words”, “pictures” and “videos”. The package “controller” was for containing classes for controlling the stages respectively including main game stage controller and explanation stage controller and video stage controller. And we had a main controller for triggering the start of the game in the package “mulanGame”. The package “data” was for storing the underlying data logic and the package “tool” was for storing classes acting as tools for checking answers and getting data. The rest two packages “layout” and “style” are for storing fxml files and css files respectively. In this way, our package structure became neat and concentrated.

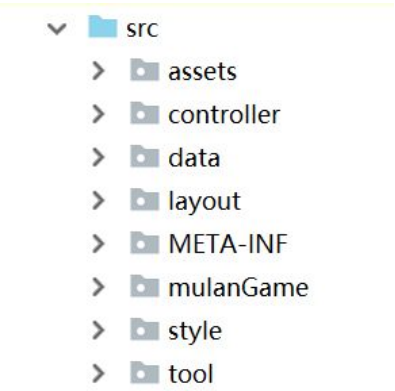


Figure 36: Our final structure of the program

6.5 Documentation

Our program was designed to have good self-documentation. From top to bottom, our packages, classes, methods and variables all had straightforward names. Methods were not with large volumes. We tried to keep all the methods short and sweet and easy to read.

Furthermore, we wrote some simple comments at the top of classes to allow others to have a glance at what this class would do.

6.6 Unit Testing

Our game was designed to be developed by using JavaFX primarily and it was not very suitable for making unit testing. We tested it through running and observing the execution results. However, our data layer had complete unit testing to help ensure that the data were well-organized and the “read” and “write” methods were correct.

6.7 Path Management and Export

The path management was tricky and troubled us for a long time, and it led to the project not exportable when we wanted to export an executable jar file. Finally, we found the solution was that using “getResource” method instead of any other methods when searching paths. IntelliJIDEA was a useful tool for exporting executable jar file and our final deliverables were a zip file of the whole structure of code and an executable jar file. The jar file can be run in two ways:

1. Double click it if the device has installed the Java Virtual Machine (JVM).

2. Run in command line by using “java -jar mulanStory.jar” under the directory it belongs to.

7. Evaluation

After finishing the project, we evaluated the overall fulfilment of our project. We want to analyze some merits and demerits of our project in this part.

Figure 37: Evaluation

7.1 Project Evaluation

During the whole cycle, our team members all communicated and cooperated well based on respecting each other and taking the project seriously. We made detailed work plan and labor allocation at the beginning of the project. And during the whole cycle, we strictly enforced the plan and eventually delivered all deliverables successfully and timely. This owes the positive participation of all the team members and making good use of some useful tools.

The deliverable lists:

1. A zip file wrapped all programming files in it which were all written by ourselves.

2. A executable jar file which was exported from the IDE IntelliJIDEA.

3. A report that used to report our experience in this project.

7.2 Product Evaluation

7.2.1 Design Evaluation

According to the user demand, our game was designed to be an educational one facing teenagers in computer science field. The game was in the form like a well-known game called “Scratch”. We have combined fascinating story with Java knowledge and embedded many exquisite images and media in the game to make learning process more interesting because Java is really complicated. The advantage of our game is that it is very educational with a bunch of Java knowledge embedded in it, which meets the user demand tightly.

However, our game is a little bit complicated as it contains some tricky Java knowledge points. This is destined at the beginning when we chose to cope with Object Oriented Programming.

7.2.2 Development Evaluation

We developed the game with the MVC pattern for separating the three concerns from each other to make the implementation light and sweet. We drew lessons from the Db coursework to construct neat underlying data structure to make the use of data in a more elegant way. We stuck to good programming manners such as making unit testing and using callback methods to add events to widgets. However, we still have many points that can be improved.

(1) There pops up a window for pushing forward the story by playing videos in it. After the players close the window by hand, the video is still playing in the background.

(2) As the components in the layout were designed to be flexible according to the length of contents they contained, the labels might have changes in lengths and sometimes labels would overlap.

(3) “reset” button is for resetting all the labels in the library and we didn’t set “undo” button for undoing the last action.

(4) We only added button for switching to the previous chapter, without skip chapters. In fact, the data of the process of the game was not stored.

(5) The answers were not order-sensitive.

7.2.3 Use Cases

We covered use cases in our evaluation by finding two friends of us, who knew something about but not proficient in Java, to play the game in order to know whether our game was user-friendly, the level design was reasonable and educational.

1. The First Use Case

The first user gave us feedback that he didn't pay attention to the pictures for hints on the left-bottom corner at first. He thought the pictures should be made more obvious. He was stuck in Chapter six, where polymorphism (upcasting) was the knowledge point embedded. What impressed him was that in chapter four for introducing encapsulation, he was misled by the dynamic picture on the left-corner and dragged "makeCheatSheet" to the answer sheet. After reading the tutorial, he solved the problem. We deliberately set the game in this misleading way so that users could be impressed and remembered the knowledge points firmly after they read the tutorials.

2. The Second Use Case

The second user was a careful one that he always studied the explanations carefully before pressing "submit". This really helped him give the correct answers. What impressed him most was the movie clips used for pushing forward the story between every two chapters. He said he was moved and would like to watch the whole movie. This was unexpected. We aimed that our game would inspire his interests in learning Java :)

8. Conclusion

Generally, we met our aims and objectives. By great teamwork, we successfully developed the Mulan game in a short-term period. This object-oriented programming education game has a well-organized structure as well as a friendly user interface for interaction. With an interesting story, we confidently believe it would attract the player much.

We as a team not only showed the advanced programming skills but also demonstrates the teamwork spirits. Overall, we achieve the objectives below:

- Developed the technical coding ability, not just through the work on the Java Programming unit, but from individual and group efforts within this project
- Gained experience using IDE like IntelliJ IDEA
- Learned a widely used of JavaFX library
- Implement agile development in practice
- Knowing completely of a software lifecycle
- Learned how to use a version control system collaboratively
- Learned how to design and implement a piece of software in an 'object-oriented' way
- Learned how to run a team effectively and co-work effectively
- Learned some effective models of cohesive teamwork
- Learned what needs in a game

Lastly, although we have finished our game project, it is not the end. In this game, it only has one storyline, the Mulan story, so we hope that we can add more interesting storyline in the future to attract a player with different interest. Moreover, we also need to polish the game interface more sophisticatedly so that makes it seems more friendly and humanized. And it is important to fix the bugs as well, otherwise, it may affect the satisfaction of the players. Generally, we have those future development issues below:

- More fascinating story
- More levels and chapters
- Bugs fixing
- UI polishing
- Introduce more knowledge about computer science

9. References

[1] Mayer, R.E., 2016. What Should Be the Role of Computer Games in Education?. *Policy Insights from the Behavioral and Brain Sciences*, *3*(1), pp.20-26.

[2] Gavalas, D. and Economou, D., 2011. Development platforms for mobile applications: Status and trends. *IEEE software*, *28*(1), pp.77-86.

[3] Taivalsaari, A., Mikkonen, T., Ingalls, D. and Palacz, K., 2008. Web browser as an application platform: The lively kernel experience.

[4] Maloney, J., Resnick, M., Rusk, N., Silverman, B. and Eastmond, E., 2010. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), p.16.

[5] Foster, I., Zhao, Y., Raicu, I. and Lu, S., 2008, November. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*(pp. 1-10). Ieee.

[6] Cockburn, A., 2002. *Agile software development* (Vol. 177). Boston: Addison-Wesley.

[7] Boehm, B.W., 1988. A spiral model of software development and enhancement. *Computer*, *21*(5), pp.61-72.

[8] Balaji, S. and Murugaiyan, M.S., 2012. Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, *2*(1), pp.26-30.

[9] Krasner, G.E. and Pope, S.T., 1988. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, *1*(3), pp.26-49.

Web ref1. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239067/SECONDARY_national_curriculum_-_Computing.pdf.

Web ref2. https://www.tynker.com/.

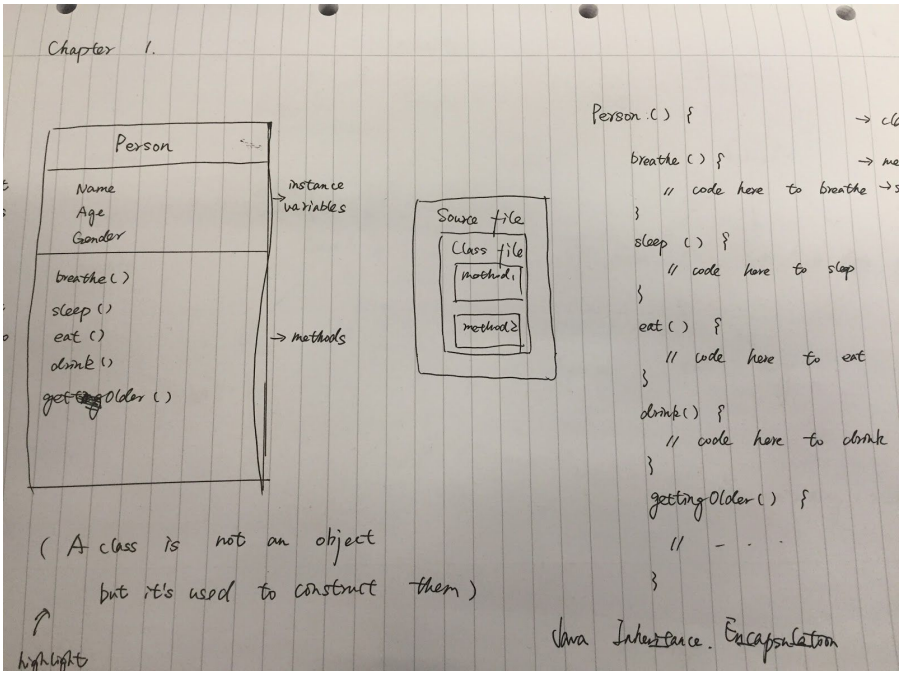
Web ref3. https://en.wikipedia.org/wiki/Scratch_(programming_language).

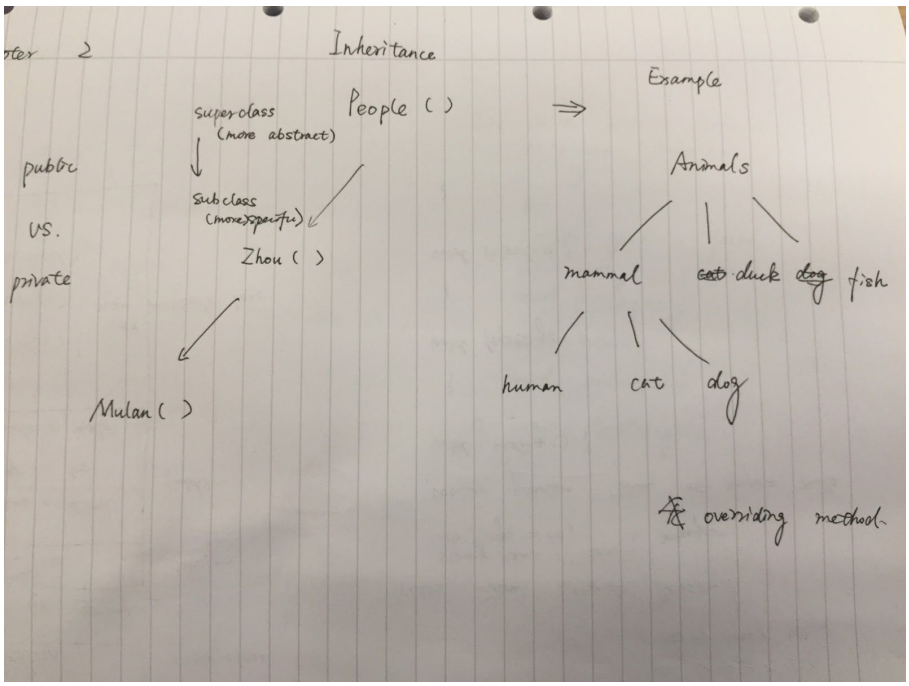
Web ref4. https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.0

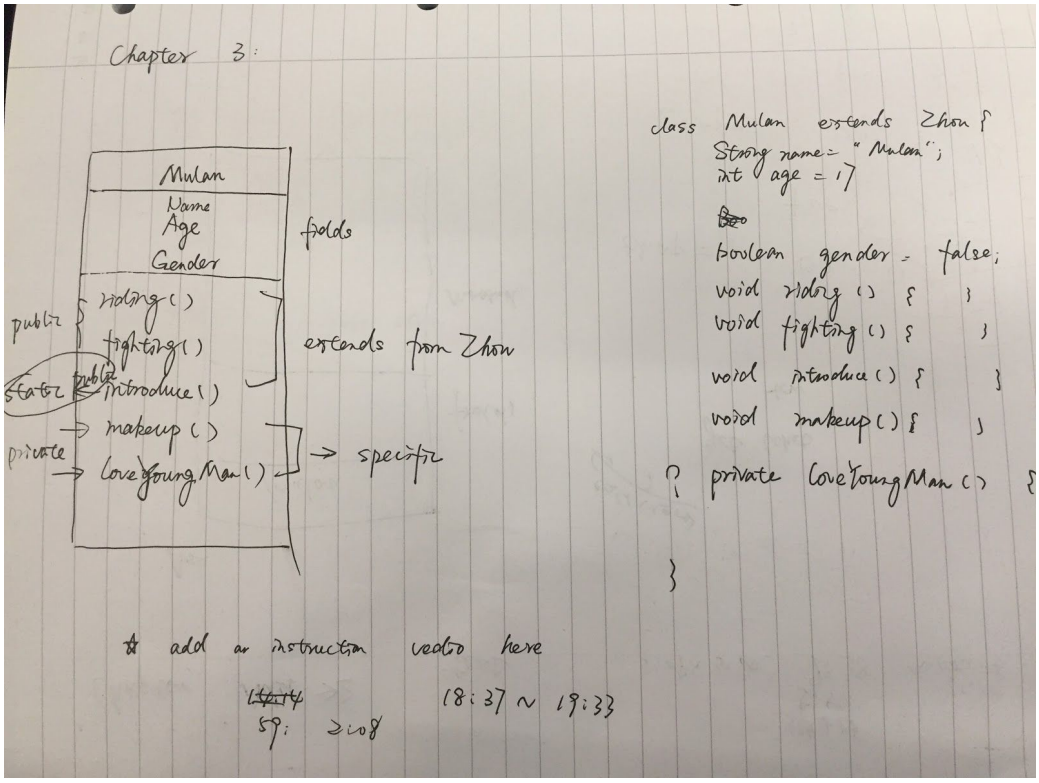
10. Appendix

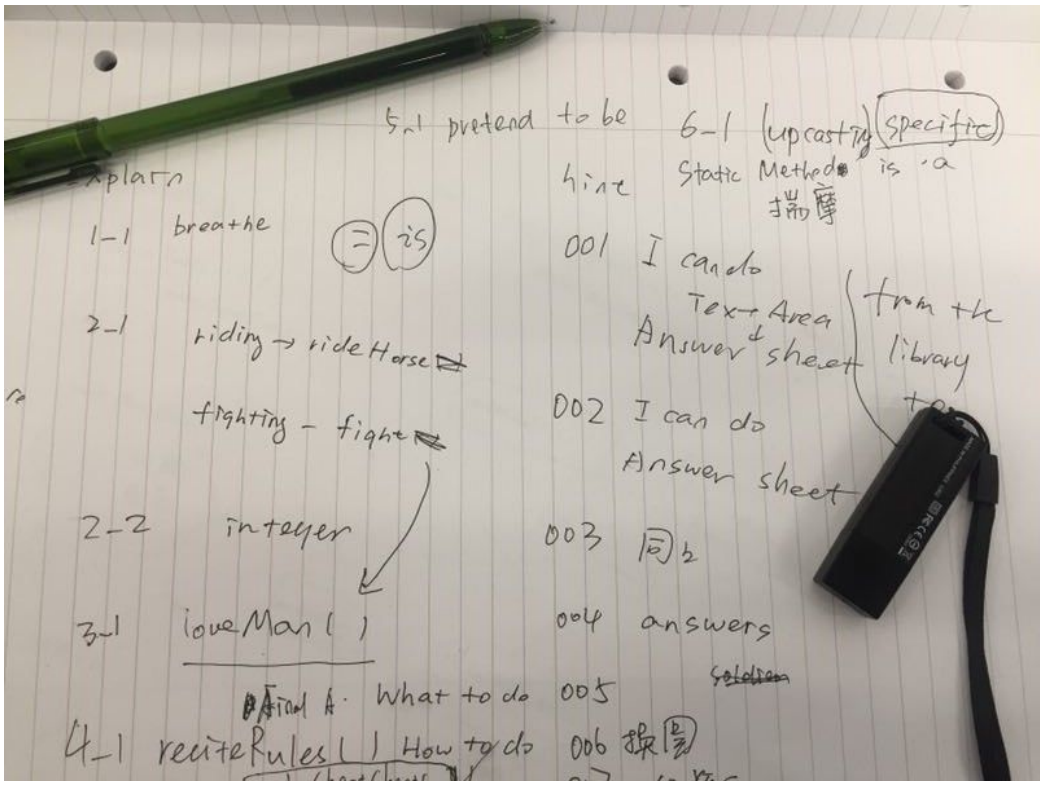
10.1 Product design 1.0

Basic Java explain add in the tutorial of this game.






Chapter 3:
Mulan
Name
Age
Gender
folds
public
riding()
fighting()
extends from Zhou
introduce()
makeup()
private
loveYoungMan()
specific
class Mulan extends Zhou {
String name = "Mulan";
int age = 17
boolean gender = false;
void riding() { }
void fighting() { }
void introduce() { }
void makeup() { }
private loveYoungMan() {
}
add an instruction redo here
18:37 ~ 19:33


plan
1-1 breathe
2-1 riding → rideHorse
fighting - fight
2-2 integer
3-1 loveMan()
4-1 reciteRules() How to do
5-1 pretend to be
hire
6-1 upcasting
specific
is-a
Static Method
001 I can do
Text Area
Answer sheet
002 I can do
Answer sheet
003
004 answers
005
006
from the library

Hints : Create Human

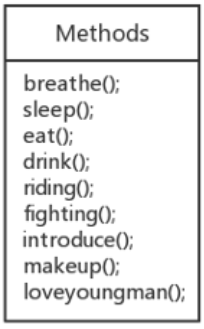
```
class Human {
    breathe();
    sleep();
    eat();
    drink();
    gettingolder();
}
```

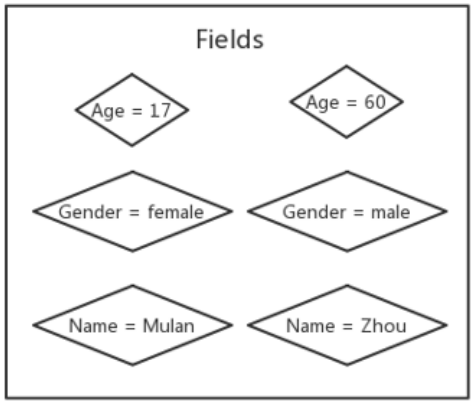
Hints : Create Mulan(Zhou's daughter)

```
class Mulan extends Zhou{
    Age = 17;
    Gender = female;
    Name = "Mulan";
    riding("Mulan is riding");
    fighting("Mulan is fighting");
    introduce();
    makeup();
    loveyoungman();
}
```

Hints : Create Zhou (Zhou is an old Human)

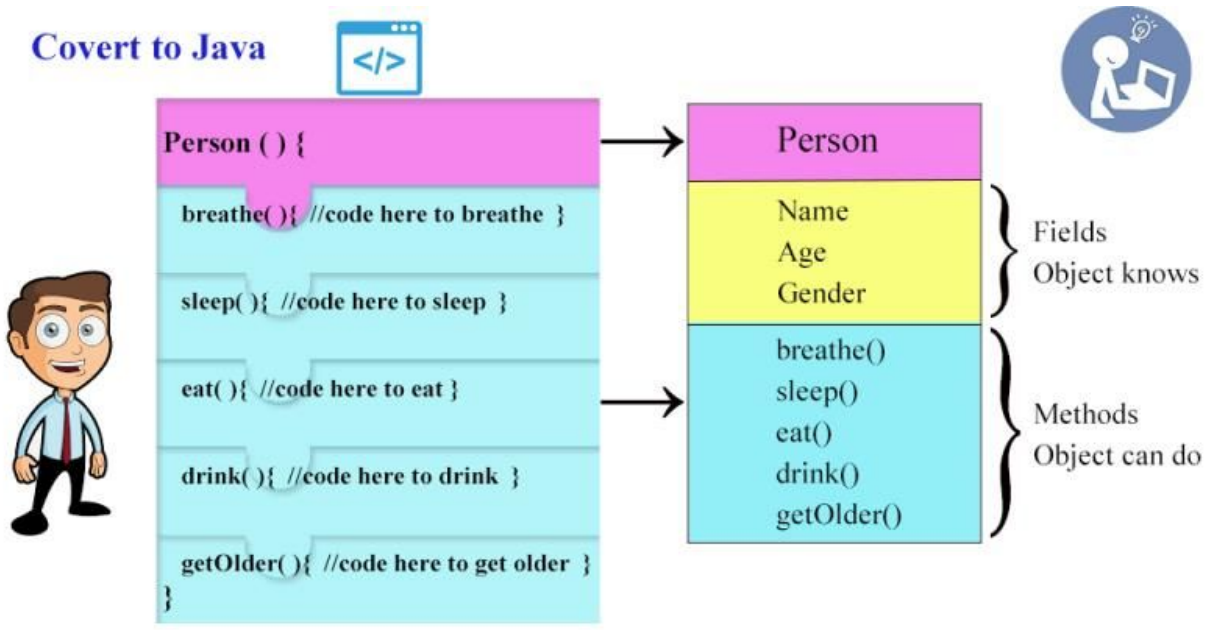
```
class Zhou extends Human{
    Age = 60;
    Gender = male;
    Name = "Zhou";
    riding("Zhou is riding");
    fighting("Zhou is fighting");
    introduce();
}
```

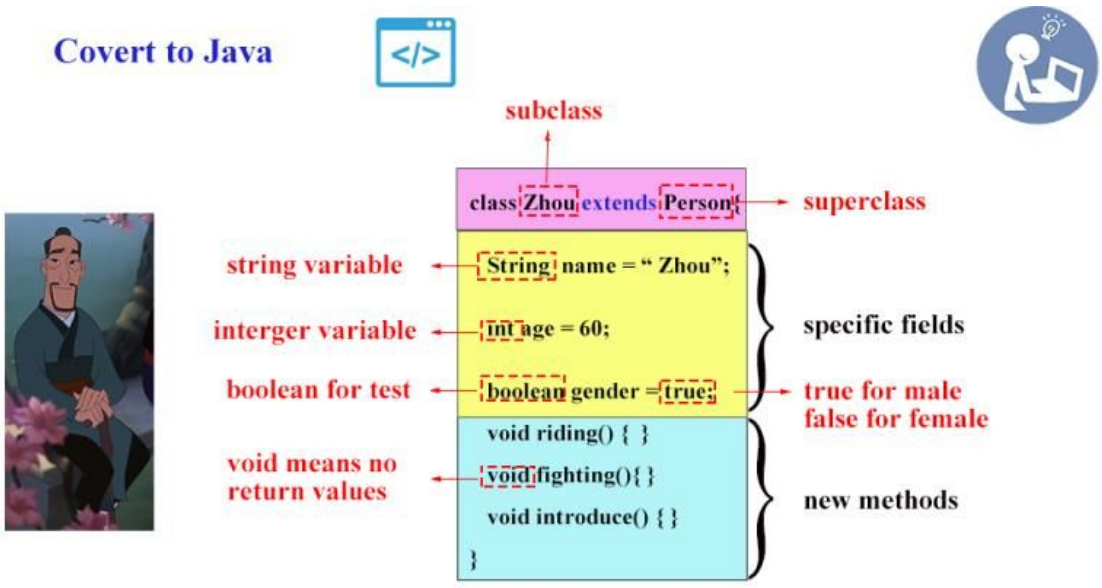


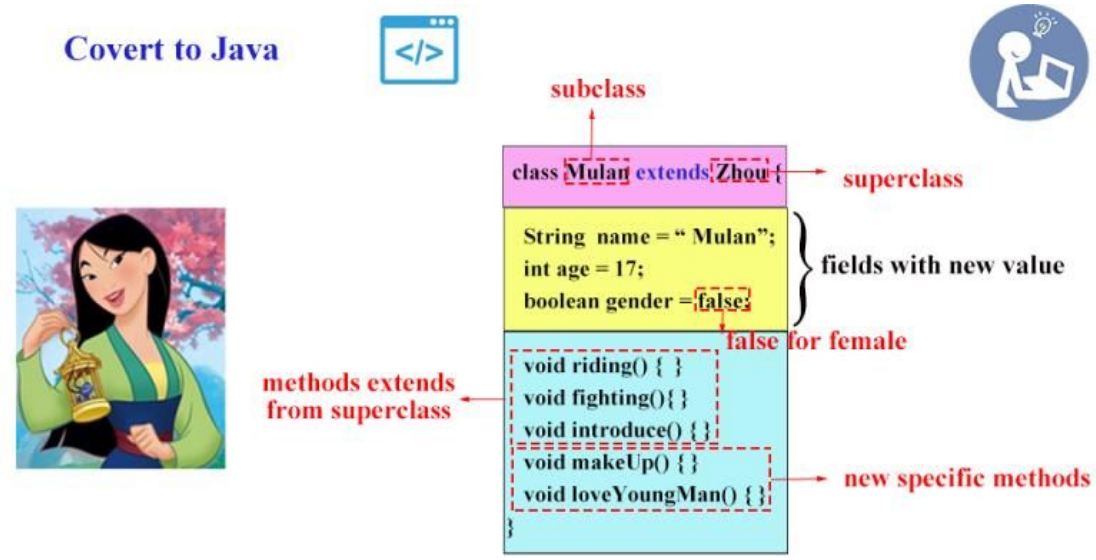


10.2 Product design 2.0

In this version, in order to introduce this Java explain tutorial more clearly, we add a convert the concept to Java code part and list these code in colourful list boxes.


Covert to Java
Person ( ) {
breathe( ){ //code here to breathe }
sleep( ){ //code here to sleep }
eat( ){ //code here to eat }
drink( ){ //code here to drink }
getOlder( ){ //code here to get older }
}
Person
Name
Age
Gender
Fields
Object knows
breathe()
sleep()
eat()
drink()
getOlder()
Methods
Object can do





10.3 Product design 3.0 (Final Version)

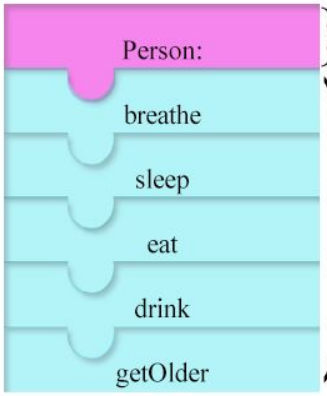
In the final version, we exchange the convert part to real piece of code and add more explain tips in the tutorial.

Object Oriented Programming - Java

Chapter 1-1

Object have states(Fields) and behaviors(Methods).

Class is a blueprint for an Object.



Class

Methods
(Object does)

Three main features of Java

- Inheritance
- Encapsulation
- Polymorphism

Covert to Java

Chapter 1-2

method with no return value

single line comment:
The compiler ignores everything
from // to the end of the line.

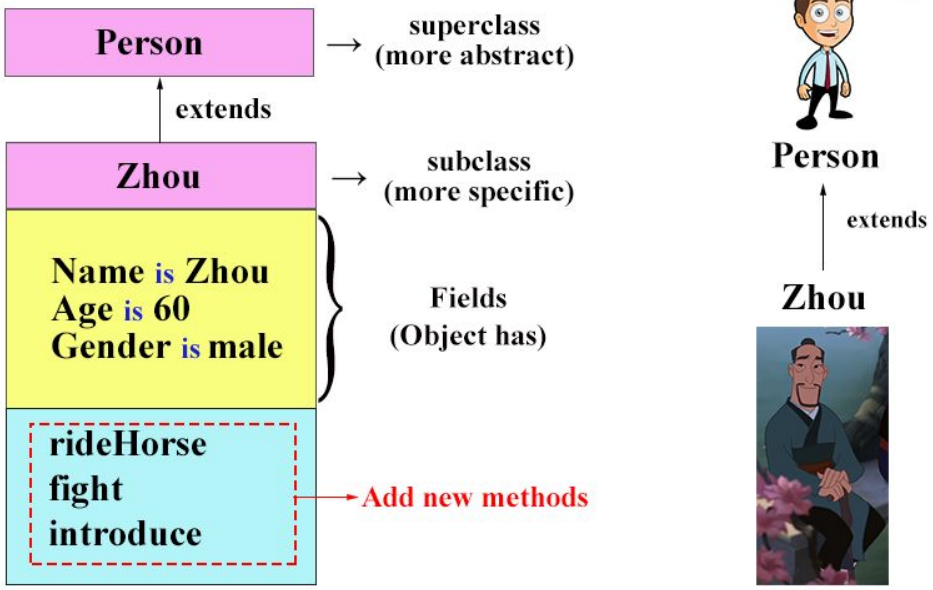
```
1  class Person{
2      void breathe(){
3          //code here to breathe
4      }
5      void sleep(){
6          //code here to sleep
7      }
8      void eat(){
9          //code here to eat
10     }
11     void drink(){
12         //code here to drink
13     }
14     void getOlder(){
15         //code here to get older
16     }
17 }
```

class name

Methods
(Object can do)

Inheritance

Chapter 2-1



Covert to Java

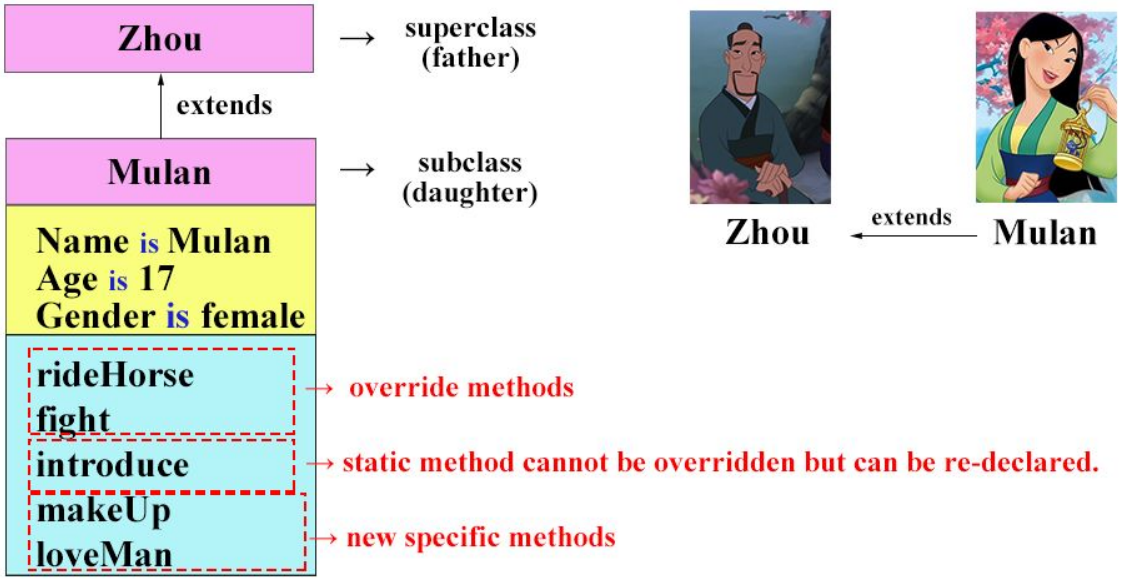
Chapter 2-2

```
1  class Zhou extends Person{
2      String name = "Zhou";
3      int age = 60;
4      boolean gender = true;
5
6      void rideHorse(){
7          //code here to ride horse
8      }
9      void fight(){
10         //code here to fight
11     }
12     static void introduce(){
13         //code here to introduce
14     }
15 }
```

subclass
superclass
string variable
integer variable
boolean for test
true or false
specific fields
true for male
false for female
new methods
Static method is accessible to every instance of a class.

Inheritance

Chapter 3-1



Covert to Java

Chapter 3-2

```
1  class Mulan extends Zhou {
2      String name = "Mulan";
3      int age = 17;
4      boolean gender = false;
5
6      @Override
7      void rideHorse(){
8          //code here to ride horse
9      }
10     @Override
11     void fight(){
12         //code here to fight
13     }
14     static void introduce(){
15         //code here to introduce
16     }
17     void makeUp(){
18         //code here to make up
19     }
20     void loveMan(){
21         //code here to love man
22     }
23 }
```

subclass

superclass

fields with new value

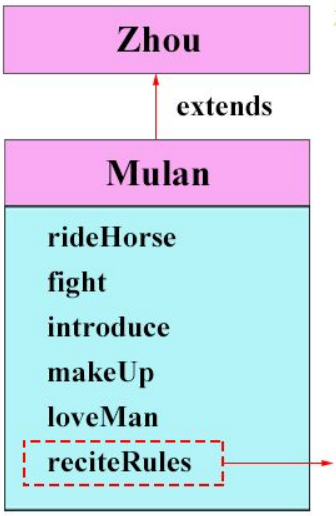
override methods

re-declared static method

new specific methods

Encapsulation

Chapter 4-1



Encapsulation used to hide the internal representation.
Only knows What to do instead of How to do.

Bad behaviour

Call this method without know the cheat sheet in it.

Covert to Java

Chapter 4-2

```
1  class Mulan extends Zhou {
2      String name = "Mulan";
3      int age = 17;
4      boolean gender = false;
5
6      //other methods...
7
8      public String reciteRules(){
9          String cheatSheet = "right answers";
10         return cheatSheet;
11     }
12 }
```

public means can be called by any object

type of method

cheat sheet in it

return the same type of method

Polymorphism (Upcasting)

Chapter 5-1

Poly = "many" morph = "form"

take my father's place

soldierMulan is a Object of Mulan upcasting to Zhou

soldierMulan
Name is Zhou Age is 60 Gender is male

} fields same as Zhou

Zhou ← pretend ← soldierMulan

Polymorphism upcasting: From subclass to superclass

Covert to Java

Chapter 5-2

upcasting from Mulan to Zhou

boolean for test true or false

to print out in Java

throw Error for test

not b

test method

```
1  Zhou soldierMulan = new Mulan();
2
3  void claim(boolean b){
4      if (b)  System.out.println("Test success");
5      if (!b) throw new Error("Test failure");
6  }
7
8  claim(soldierMulan.name.equals("Zhou"));
9  claim(soldierMulan.age == 60);
10 claim(soldierMulan.gender == true);
```

Call the test method *claim* can know that the fields in soldierMulan are same as Zhou.

polymorphism upcasting: Fields are same as superclass.

Polymorphism (Upcasting)

Chapter 6-1

Poly = "many" morph = "form"

Override Methods are same as subclass unless the method is static.

Covert to Java

Chapter 6-2

```
1  //superclass
2  class Zhou extends Person{
3      void rideHorse(){ }
4      void fight(){ }
5      static void introduce(){ }
6  }
```

```
1  //subclass
2  class Mulan extends Zhou {
3      @Override
4      void rideHorse(){ }
5      @Override
6      void fight(){ }
7      static void introduce(){ }
8      void makeUp(){ }
9      void loveMan(){ }
10 }
```

call the override methods in Mulan

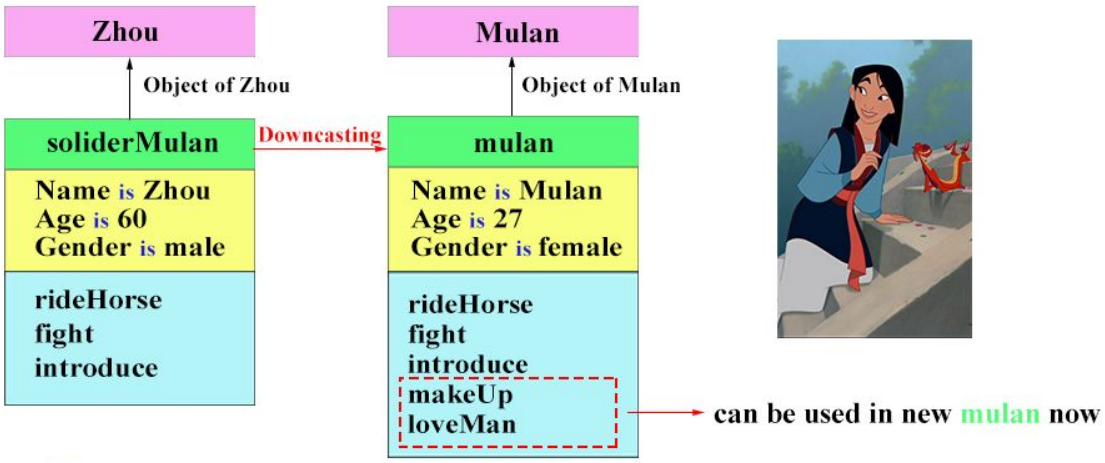
```
1  Zhou soldierMulan = new Mulan();
2
3  soldierMulan.rideHorse();
4  soldierMulan.fight();
5  soldierMulan.introduce();
```

call method in Zhou because it is static

specific methods in Mulan cannot be used.

Polymorphism (Downcasting)

Chapter 7-1



Use downcasting when we want to access specific methods of a subclass.

Covert to Java

Chapter 7-2

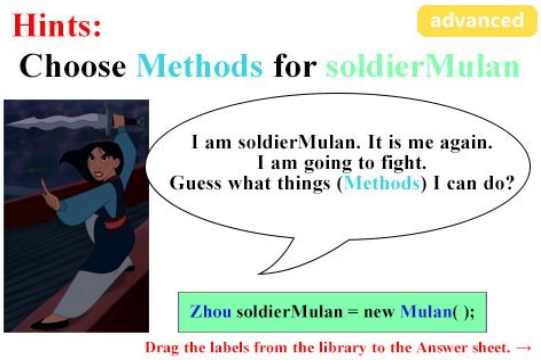
```
1  //10 years later
2  Mulan mulan = (Mulan)soldierMulan;
3
4  void claim(boolean b){
5      if (b)  System.out.println("Test success");
6      if (!b) throw new Error("Test failure");
7  }
8
9  claim(mulan.name.equals("Mulan"));
10 claim(mulan.age == 27);
11 claim(mulan.gender == false);
12
13 mulan.rideHorse();
14 mulan.fight();
15 mulan.introduce();
16 mulan.makeUp();
17 mulan.loveMan();
```

Downcasting from Zhou to Mulan

extends getOlder() method in Person to get older.

All the Fields and Methods downcasting to Mulan subclass.

The Hints part which is used to lead the users to answer the chapter questions.



10.4 meeting minutes

Meeting minutes-1

Meeting minutes

Time:2018/2/5 15:00~17:00

Location: MVB atrium

• Details for cooperation

Communication: Wechat ,e-mail

Files share: One drive

Development environment: IntelliJ IDEA

Version control system: gitlab

• forms of game

Meeting minutes-1

• forms of game

Idea table

No	Idea	Concept taught	Creative and fun	Player age
1	Computer science quiz competition for children	Concept of computer science, Basic knowledge of computer architecture, Introduction To algorithm	Learn Computer science knowledge in the fun of the competition	6~18
2	Training for Object oriented programming	Concept of object oriented(Java)	Use image, sound and stories to attract children's attention	14~18

Meeting minutes-1

3	Typing game	Basic computer skills	Constant level up and give feedback	6~10
4	Sorting game	Sort algorithm	Visualize the sorting process	14~18
5	Computer structure game combination	Concept of computer science, Basic knowledge of computer architecture	Use A variety of dazzling mini games to Stimulate curiosity	10~18
6	get out of the maze	Logic, programming		10~18
8	Computer decryption game	Cryptography, algorithm	Win a reward by decryption and gain a sense of accomplishment	10~18
9	Chinese programming game	Help Chinese primary students who don't understand English learning basic computer skills	Using Mandarin to help Chinese pupils will inspire their interests	6~10

Meeting minutes-1

			accomplishment	
9	Chinese programming game	Help Chinese primary students who don't understand English learning basic computer skills	Using Mandarin to help Chinese pupils will inspire their interests	6~10

• Presentation Planning

Final selection: idea2 and idea5.

PPT Outline: inspiration, aim, tool, creative thinking.

• To do

1.Improve idea table (especially no.6).

2.Come up with ideas for presentation.

3.learn how to use git.

Meeting minutes-2

Meeting minutes2

Time:2018/2/9 15:00~16:00

Location: Maths Building

1. More details in 2 options.
   1) Tell a story: Mulan 's story
   2) LEGO: from computer to logic gates
2. Determining deadline of presentation.

Meeting minutes-3

Time:2018/2/12 14:40~15:20

Location: MVB

1. Details
2. Project management:
   A. tasks
      a) analysis
      b) design
      c) implementation
      d) testing
      e) deployment
   B. responsibility
      a) QIAN: project planning
      b) JIA: write a script
      c) others: supporting