Assignment 2

Exercise 1

You have seen different ways of documenting the effects of a method in the lecture (e.g., checked exceptions, read and write effects). Purity annotations are a specific kind of write effect specification: a *pure* method i) does not modify any object that already existed in the pre-state (but it can modify newly created objects), and ii) for given set of arguments (including the receiver) the return value (if any) is always the same. For instance, in the code below method `equals` and method `hashcode` are probably pure, while method `getImage` is definitely not pure since it modifies the current object by initializing the image field.

```
class ImageFile {
  String file;
  Image image;

  ...

  public Image getImage() {
    if (image == null) {
      Image tmp = Image();
      // load the image
      image = tmp;
    }
    return image;
  }

  boolean equals(Object o) {
    if( o.getClass() != getClass() ) return false;
    return file.equals( ((ImageFile) o).file );
  }

  int hashcode() {
    if (image == null) {
      return file.hashcode();
```

```
    } else {
      return image.hashcode() + file.hashcode();
    }
  }
}
```

1. Why are method `equals` and method `hashcode` *probably* pure? Under which circumstances are they not pure? Is is possible to change the class design such that they are pure under all circumstances?

2. Can you think of a practical solution that would catch (at runtime) violations of the first requirement? Apply the instrumentation to the code from above.

3. How would your instrumentation deal with commonly used designs, such as lazy initialization of data structures or caching?

Exercise 2

Write down documentation in the form of contracts for the code below:

```
class ImageFile {
  private final String file;
  private Image image;

  ImageFile(String f) {
    file = f;
    image = null;
  }

  public Image getImage() {
    if (image == null) {
      Image tmp = Image();
      // load the image
      image = tmp;
    }
    return image;
  }

  public boolean equals(Object o) {
    if( o.getClass() != getClass() ) return false;
    return file.equals( ((ImageFile) o).file );
  }
```

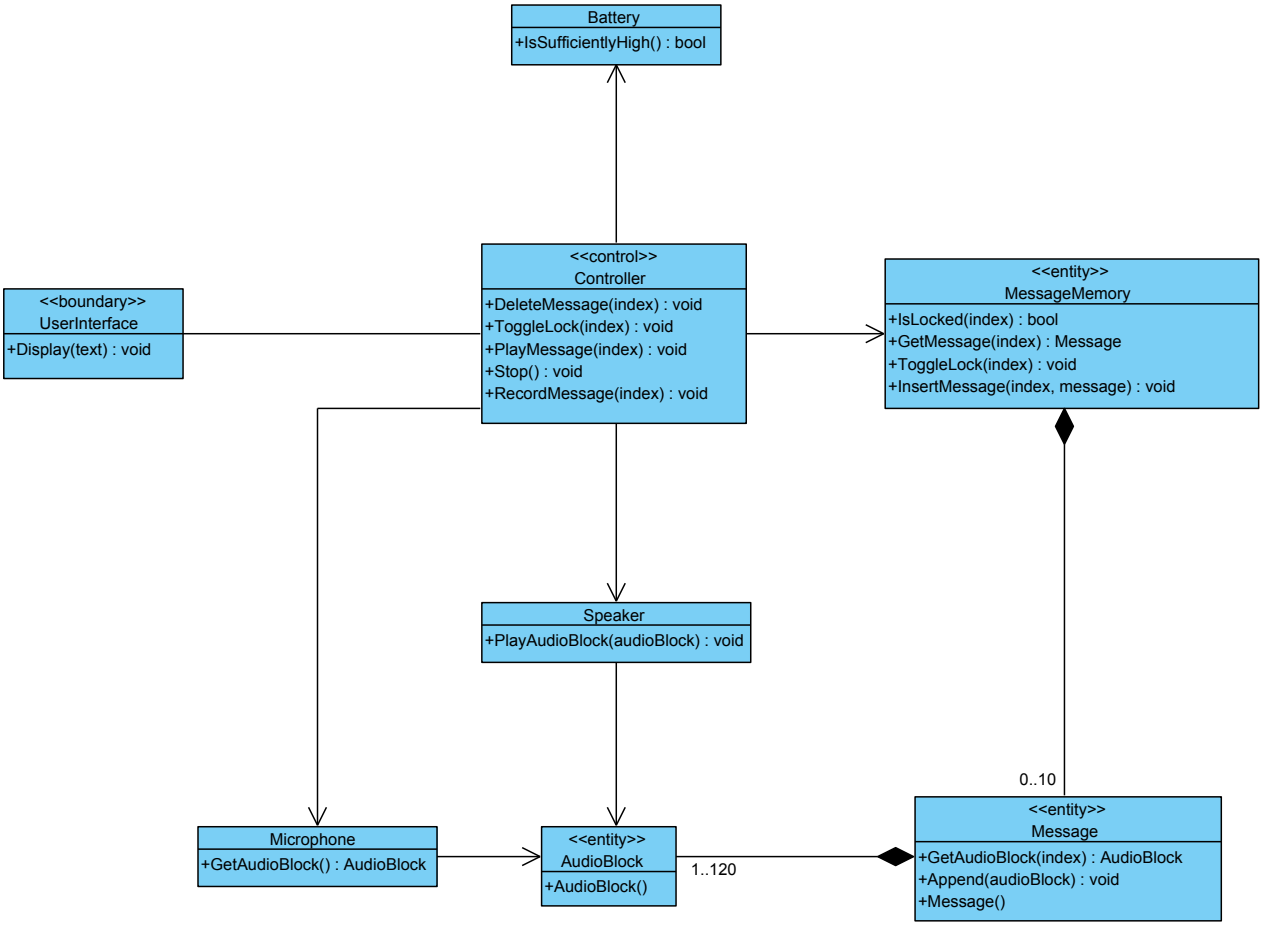
```
  public int hashcode() {
    return getImage().hashcode() + file.hashcode();
  }
}
```

Exercise 3

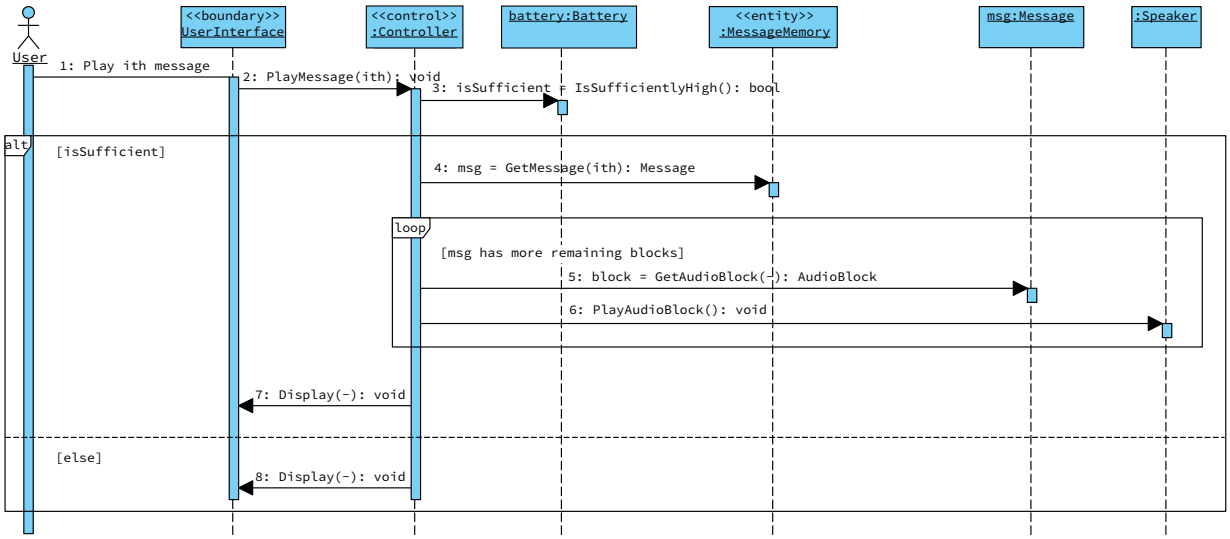
1. Draw a UML class diagram for the system described below:
   (a) every student is either undergraduate or graduate student. No student is both undergraduate and graduate student;
   (b) a student should register at a university, and only registered students are legal students;
   (c) every student has a unique student ID, and he or she has only one major;
   (d) students with the same major are regarded as classmates, students can have several classmates.
2. Which properties of the system above cannot be captured using UML class diagrams?

Exercise 4

You are given a class diagram for the *Dictaphone* system below:



1. Read the sequence diagram below and write down pseudo-code for method `PlayMessage`.



2. Write down a sequence diagram for the following use cases:

   - Use case 1: Delete message

User	System
1. User asks the system to delete the i-th message.	2. The system checks if the system is locked (extension point). 3. Message is not locked, it deletes the message and notifies the user.

   - Use case 2: Fail to delete message (extends use case 1)

User	System
	3. Message is locked and the system displays an error to the user.