Assignment 2 (Solution)

Exercise 1

1. They are only pure if its callees don't modify any objects that existed in the prestate of the caller.

2. One could keep track of newly allocated objects within each method body and check that only those objects are modified.

```
class PurityChecks
{
  static IdentityHashMap<object, Boolean> fresh;
  static boolean checking;
}
class ImageFile
{
  private String file;
  private Image image;

  public void setFile(String file)
  {
    + assert !PurityChecks.checking || PurityChecks.fresh.containsKey(this);
    this.file = file;
    var i = new Image(file);
    + PurityChecks.fresh.put(i, true);
    + assert !PurityChecks.checking || PurityChecks.fresh.containsKey(this);
    this.image = i;
  }

  @Pure
  boolean equals(Object o) {
    + var oc = PurityChecks.checking;
    + PurityChecks.checking = true;
    + var of = PurityChecks.fresh;
    + PurityChecks.fresh = new IdentityHashMap<object, Boolean>();
    boolean result;
```

```
    if (!(o isinstanceof ImageFile))
      result = false
    else
      result = file.equals(((ImageFile) o).file);
    + PurityChecks.checking = oc;
    + of.putAll(PurityChecks.fresh);
    + PurityChecks.fresh = of;
    return result;
  }

  @Pure
  int hashcode() {
    + var oc = PurityChecks.checking;
    + PurityChecks.checking = true;
    + var of = PurityChecks.fresh;
    + PurityChecks.fresh = new IdentityHashMap<object, Boolean>();
    int result;
    if (image == null)
      result = file.hashcode();
    else
      result = image.hashcode() + file.hashcode();
    + PurityChecks.checking = oc;
    + of.putAll(PurityChecks.fresh);
    + PurityChecks.fresh = of;
    return result;
  }
}
```

3. For such designs, the instrumentation would report errors even if the effects can't be observed by a client. We could weaken the definition of purity for such designs.

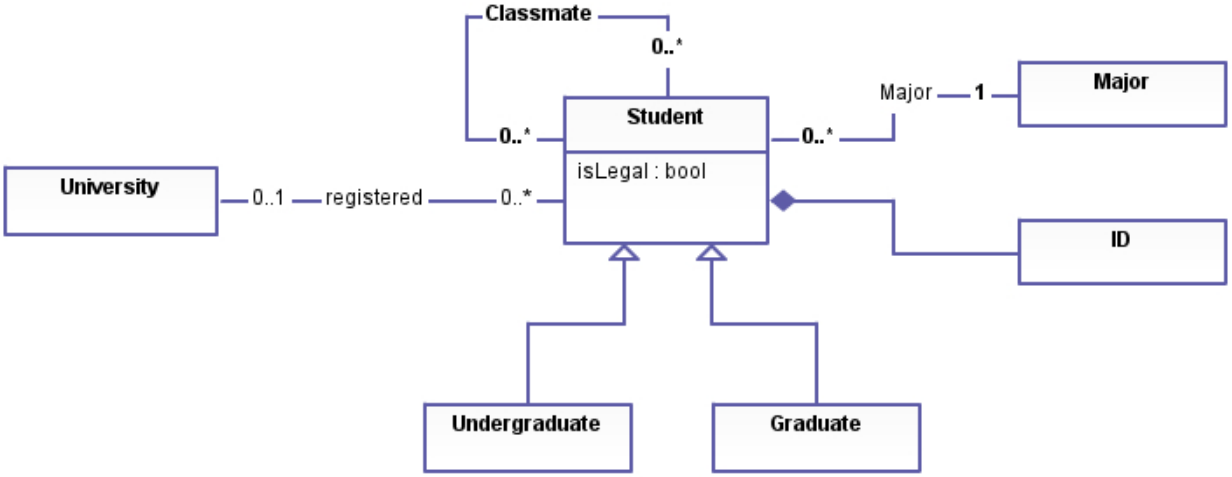
Exercise 2

Contracts:

1. `file` is non-null (class invariant)
2. `getImage()` returns non-null (post-condition)
3. Object `o` is non-null (pre-condition for `equals`)
4. String `s` is non-null (pre-condition for constructor of class `ImageFile`)

Exercise 3

1. Student Class diagram



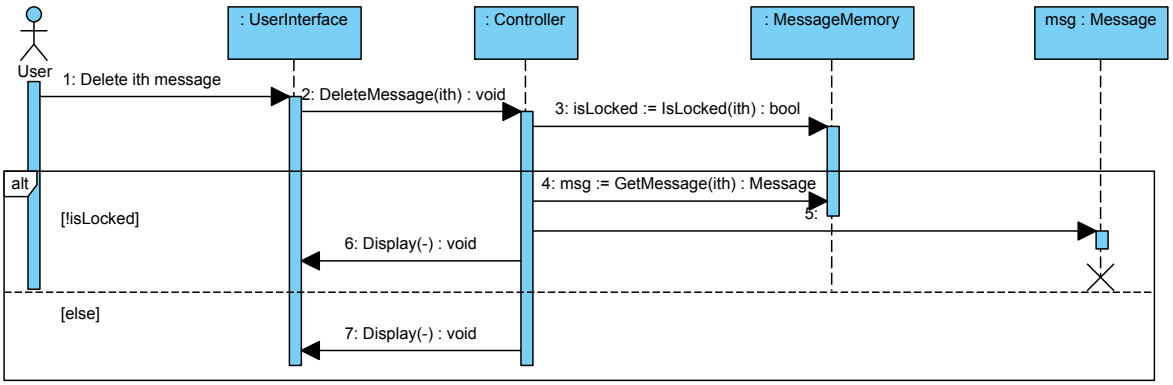
2. (a) Uniqueness of ID
   (b) Classmates have the same major
   (c) A student is legal iff he/she is registered

Exercise 4

1. Pseudo-code for method `PlayMessage`:

```
if Battery.isSufficientlyHigh() then
  message = MessageMemory.GetMessage(i)
  for each block index j in message do
    block = message.GetAudioBlock(j)
    Speaker.PlayAudioBlock(block)
  UserInterface.Display("message played")
else
  UserInterface.Display("battery low")
```

2. Sequence diagram:


User
: UserInterface
: Controller
: MessageMemory
msg : Message
1: Delete ith message
2: DeleteMessage(ith) : void
3: isLocked := IsLocked(ith) : bool
alt
4: msg := GetMessage(ith) : Message
5:
[!isLocked]
6: Display(-) : void
[else]
7: Display(-) : void