OOSE Assignment 2

1. Use cases

Title

Selecting a piece

Actors

The current player

Pre-condition

The game has not ended.

Main path

1. The player clicks on one of the Brickus pieces on the bottom pane of the game window.
2. The piece clicked is highlighted. After a piece is selected, when the player's mouse cursor enters the game board, the cursor becomes the piece selected.

Title

Rotating a piece

Actors

The current player

Pre-condition

The game has not ended. The player has selected a piece

Main path

1. The player rotates the mouse wheel in the up/down direction by 1 unit
2. The piece selected rotates clockwisely/counter-clockwisely by 90 degrees.
3. The selected piece in the bottom pane of the game window and the piece representing the mouse cursor in the game board update their graphics when the player rotates the piece.

Title

Flipping a piece

Actors

The current player

Pre-condition

The game has not ended. The player has selected a piece

Main path

1. The player clicks the mouse's right button, the piece selected flips horizontally; if the player clicks the mouse's right button while holding the shift key, the piece selected flips vertically.
2. The selected piece in the bottom pane of the game window and the piece representing the mouse cursor in the game board update their graphics when the player flips the piece.

Title

Skipping the turn

Actors

The current player

Pre-condition

The game has not ended.

Main path

1. During a player's turn, the player can click the "pass" button.
2. If the opponent just ended his/her turn by placing a piece, the current player's turn ends. The pieces in the bottom pane are replaced by the remaining pieces of the other player and no piece is highlighted in the pane and the mouse cursor does not show the shape of any piece.

Alternative path

2.1. If the opponent just clicked "pass" to skip his/her turn, the game ends. The bottom pane of the remaining pieces is removed from the window, the "pass" button is disabled, and the cursor does not show the shape of any piece. If the two players have the same score, the status bar shows the message "Game over. Game tied."; if the two player have different scores, the status bar shows "Game over. The winner is Player 1" if player 1 has higher score and vice versa.

Title

Placing a piece

Actors

The current player

Pre-condition

The player has selected a piece

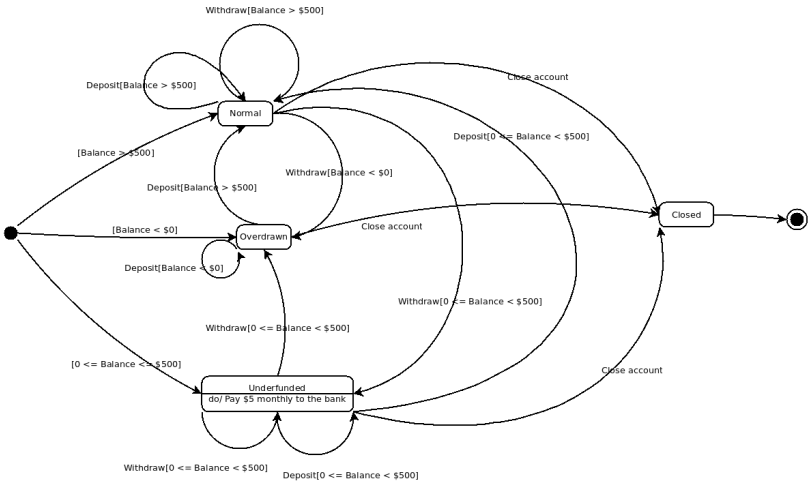
Main path

1. The player can rotate and flip the selected piece or select another piece or choose to skip the turn at any time before his/her turn ends.
2. The player moves the mouse cursor to the position that he/she wants the piece to be placed.
3. The player clicks the left mouse button.
4. If the rules of Brickus allows this move, the piece is shown on the position the player chooses and disappears from the bottom pane of the window. The player's score on the status bar updates according to the number of grids of the placed piece. The pieces in the bottom pane are replaced by the remaining pieces of the other player and no piece is highlighted in the pane and the mouse cursor does not show the shape of any piece. The current player's turn ends.

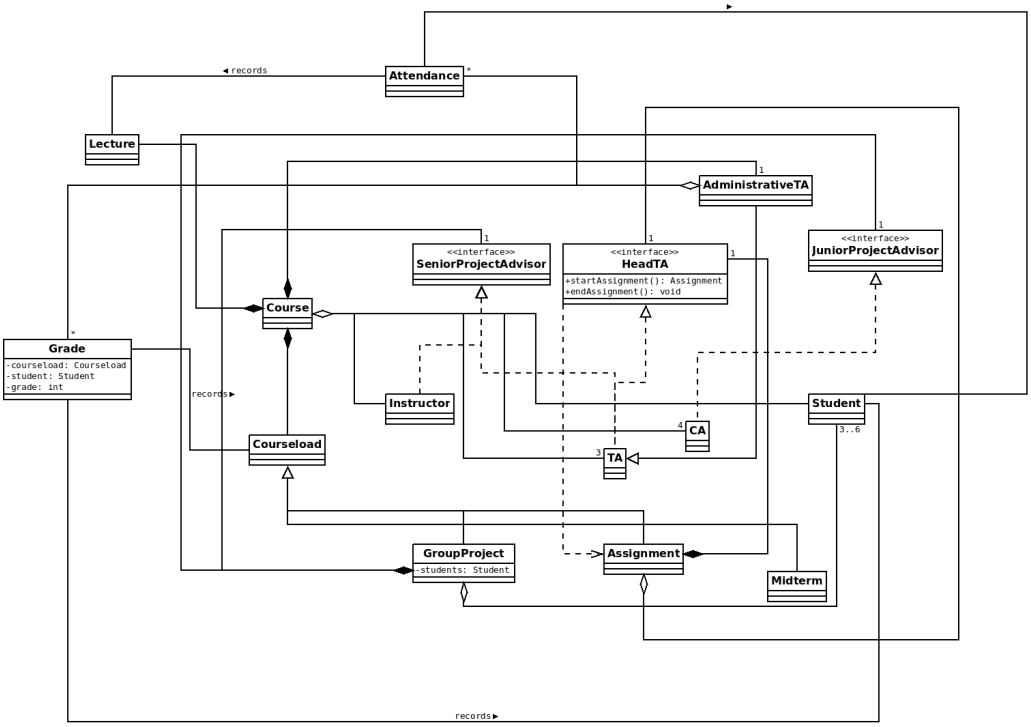
Alternative path

4.1. If the rules of Brickus do not allow this move, the player cannot place the piece on the board and the status bar shows the message explaining why this move is invalid. The player needs to continue to place a piece or skip the turn.

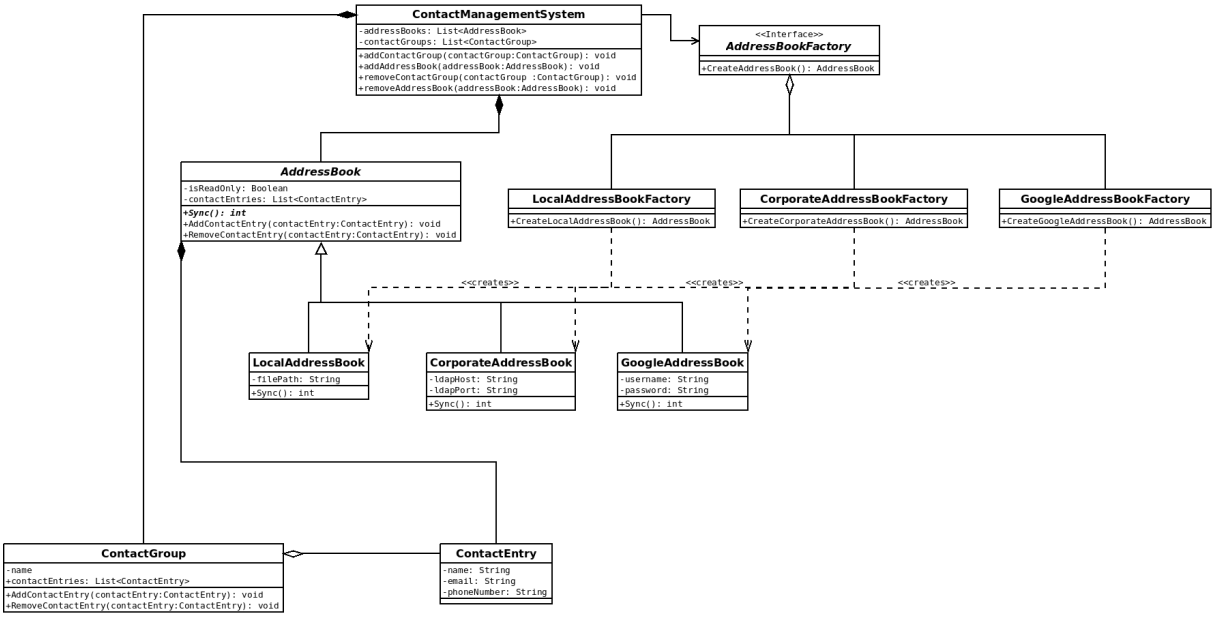
2. State diagram



3. Domain model



4. Class diagram



5. Software principles

This snippet violated OCP. The `ResourceAllocator` method is not open for extension. If we want to add in the support for another type of resource, we need to write two more `case` statements. We can use a factory method here.

```
abstract class ResourceAllocator extends Object {
  public abstract void allocate();
  public abstract void deallocate();
}

class FileResourceAllocator extends ResourceAllocator {
  public void allocate() {
      return allocateNewFileResource();
  }
  public void deallocate(Resource resource) {
    releaseFileResource(resource);
  }
}
class MemoryResourceAllocator extends ResourceAllocator {
  public void allocate() {
      return allocateNewMemoryResource();
  }
  public void deallocate(Resource resource) {
    releaseMemoryResource(resource);
  }
}
class SocketResourceAllocator extends ResourceAllocator {
  public void allocate() {
      return allocateNewSocketResource();
  }
  public void deallocate(Resource resource) {
    releaseSocketResource(resource);
  }
}
```

It violated DRY. The code used to send the text to through all the sockets is repeated for 3 times. Write it in a separate method.

```
public class ConversationWindow extends JPanel {
    private List<Socket> sockets;

    private JTextField messageBox;

    private JButton sendButton;

    private JMenuItem sendMenuItem;

    private void sendText() {
      for (Socket s : this.sockets) {
        try {
          OutputStreamWriter write = new OutputStreamWriter(s.getOutputStream());
          writer.write(this.messageBox.getText());
        } catch (Exception ex) {
          /* -- Error Handling -- */
        }
      }
    }
    public ConversationWindow() {

        this.sendMenuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
              sendText()
            }
        });

        this.sendButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
              sendText()
            }
        });

        this.messageBox.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                if ( e.getKeyCode() == KeyEvent.VK_ENTER && e.isShiftDown() )
                {
                  sendText();
                }
            }
        });
    }
}
```

This violates LSP. Even if `Set` *is a* `MyMultiSet`, the `count` method makes no sense for a set. We'd rather say if the set contains a element than counting the number of the occurrence of the element. So we use delegation instead of inheritance here.

```
interface Countable<T> {
  public int count(T v);
}
interface Collection<T> {
  public void add(T v);
  public void remove(T v);
}


class MyMultiSet implements Countable, Collection {
   private List<T> data;
```

```
    public MyMultiSet() {
      data = new LinkedList<T>();
    }
    public int count(T v) {
        int c = 0;
        for(T i : data) {
            if (v.equals(i)) c++ ;
        }
        return c;
    }
}
class Set implements Collection {
  private MyMultiSet multiSet;
  public Set() {
    multiset = new MyMultiSet();
  }
  public void add(T v) {
    if (!contains(v)) {
      multiset.add(v);
    }
  }
  public void contains(T v) {
    return multiSet.count(v) > 0;
  }
  public void remove(T v) {
    data.remove(v);
  }
}
```

This piece of code violates SRP and OCP. The `User` class should not take care of sending the emails. And the constructor should be `protected` or `private` as the `User` class is using a factory method to create an object.

```
class Mailer {
    /* Send a verification mail to the user provided address */
    public void sendVerificationEmail(String name, String email) {
        /*-- Code here --*/
    }

    /* Send a welcome mail to the user once verified */
    public void sendWelcomeEmail(String name, String email) {
        /*-- Code here --*/
    }
}

class User {
    private String name, address, email ;
    private boolean registered;
    private Mailer mailer;
    protected User(String name, String address, String email) {
        this.name = name;
        this.address = address;
        this.email = email;

        this.registered = false;
        this.mailer = new Mailer();
        this.mailer.sendVerificationEmail(name, email);
    }

    public void setRegistered(boolean registered) {
        this.registered = registered;
        if (this.registered)
          mailer.sendWelcomeEmail(this.name, this.email);
    }

    /* Create a new User */
    public static User createUser(String name, String address, String email) {
        return new User(name, address, email);
    }
    /*-- Other Code */
}
```