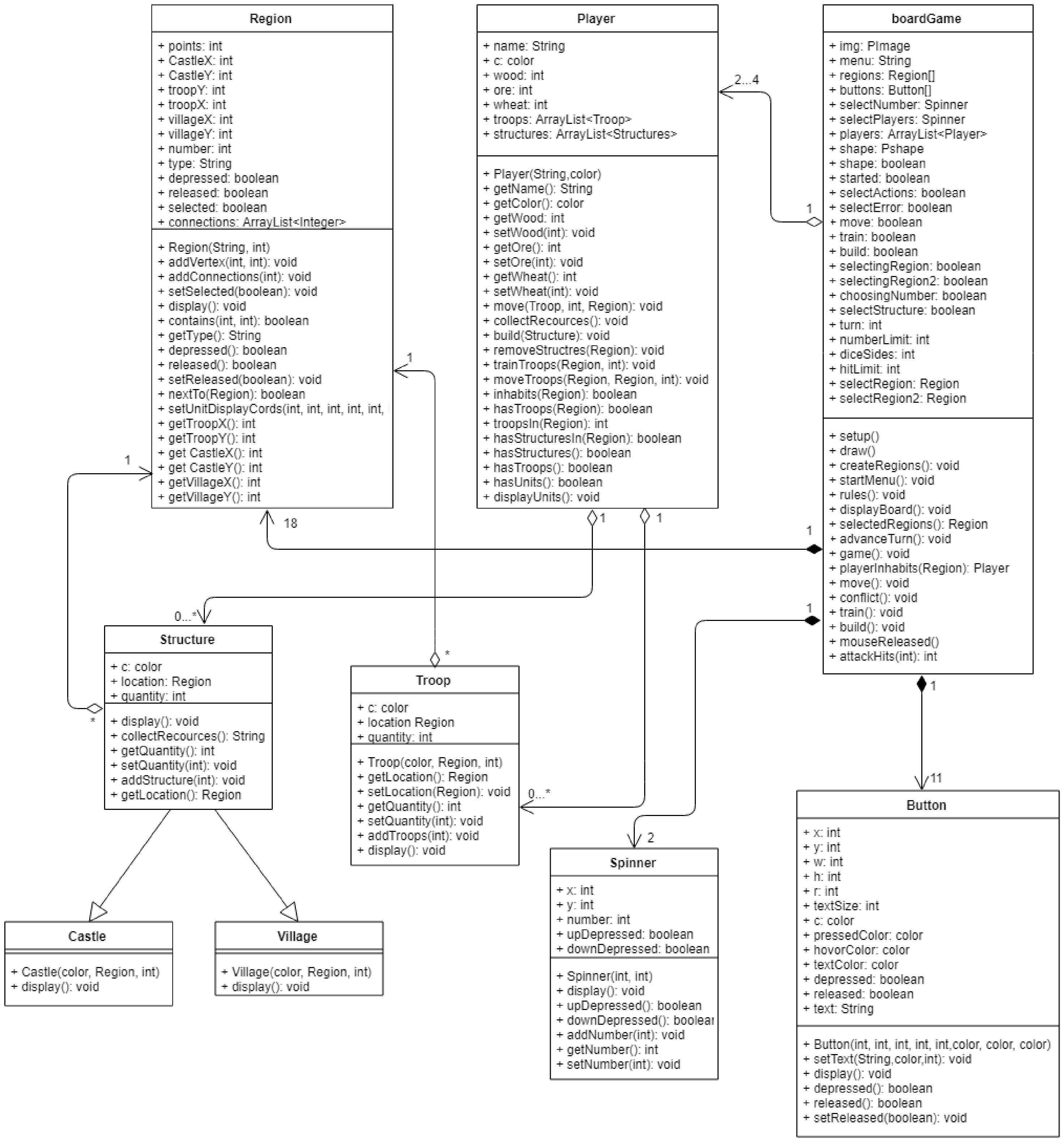
CRITERION C: Development

1. Looping nature of draw
2. Region shapes
3. Selecting Numbers
4. Displaying Units

1. I initially tried to design the game and game action function to only be called once per turn/action. However, after beginning to implementing this, I realized that this would not work with the looping nature of the draw function. In addition, in order to render the graphics and to update them, they needed to be called in a loop. Because the graphics do not re render until the draw function has looped through, I needed to design the functions so that they could be looped through continually until another function was needed. In accomplishing this I used a lot of booleans to keep track of different factors and steps at which the program was currently at.
2. In the program, the users need to be able to select regions. This was a challenge because the board I got from the makers of the game was an image. When solving this problem, I chose to use the java polygon class. I chose this because it would allow me to create a custom shape, and to be able to determine if a point is within that shape. When creating the shapes, I took coordinate points from the image and created a polygon with a large number of sides in order to best represent the lines on the board. In order to store this large number of points I placed them into a text file located in the data directory. When the program starts, the program reads the file and loads in the points into the region objects, and draws lines connecting the vertices of these polygons. I was able to manipulate the color of these lines in order to only display them when the mouse is over the region, and cause the width of the lines to increase as the mouse is pressed over that region.
3. In the program, numbers inputs need to be taken from the user. These inputs are used to control the number of players and turn actions. This is challenging in Processing due to the fact that Processing does not have a user input like Java and other languages do. To combat this, I created a number spinner class in order to control the input of numbers. The number spinner is a box that displays a number and, on the right, there is a up arrow to increment the number and a down arrow to decrement the number.
4. For displaying the units, I chose to set certain locations in each region for each type of structure, signifying the quantity by a number appearing on top of the unit. I chose this approach instead of allowing the user to place the troops because allowing the user to place the units could allow units to overlap and hide each other causing confusion. To implement this approach, I stored the coordinates of the locations the units would appear for each region in a text file. The program recorded the location just as it did with the region vertices.

UML Diagram


Region
+ points: int
+ CastleX: int
+ CastleY: int
+ troopY: int
+ troopX: int
+ villageX: int
+ villageY: int
+ number: int
+ type: String
+ depressed: boolean
+ released: boolean
+ selected: boolean
+ connections: ArrayList<Integer>
+ Region(String, int)
+ addVertex(int, int): void
+ addConnections(int): void
+ setSelected(boolean): void
+ display(): void
+ contains(int, int): boolean
+ getType(): String
+ depressed(): boolean
+ released(): boolean
+ setReleased(boolean): void
+ nextTo(Region): boolean
+ setUnitDisplayCords(int, int, int, int, int,
+ getTroopX(): int
+ getTroopY(): int
+ get CastleX(): int
+ get CastleY(): int
+ getVillageX(): int
+ getVillageY(): int
Player
+ name: String
+ c: color
+ wood: int
+ ore: int
+ wheat: int
+ troops: ArrayList<Troop>
+ structures: ArrayList<Structures>
+ Player(String,color)
+ getName(): String
+ getColor(): color
+ getWood(): int
+ setWood(int): void
+ getOre(): int
+ setOre(int): void
+ getWheat(): int
+ setWheat(int): void
+ move(Troop, int, Region): void
+ collectRecources(): void
+ build(Structure): void
+ removeStructres(Region): void
+ trainTroops(Region, int): void
+ moveTroops(Region, Region, int): void
+ inhabits(Region): boolean
+ hasTroops(Region): boolean
+ troopsIn(Region): int
+ hasStructuresIn(Region): boolean
+ hasStructures(): boolean
+ hasTroops(): boolean
+ hasUnits(): boolean
+ displayUnits(): void
boardGame
+ img: PImage
+ menu: String
+ regions: Region[]
+ buttons: Button[]
+ selectNumber: Spinner
+ selectPlayers: Spinner
+ players: ArrayList<Player>
+ shape: Pshape
+ shape: boolean
+ started: boolean
+ selectActions: boolean
+ selectError: boolean
+ move: boolean
+ train: boolean
+ build: boolean
+ selectingRegion: boolean
+ selectingRegion2: boolean
+ choosingNumber: boolean
+ selectStructure: boolean
+ turn: int
+ numberLimit: int
+ diceSides: int
+ hitLimit: int
+ selectRegion: Region
+ selectRegion2: Region
+ setup()
+ draw()
+ createRegions(): void
+ startMenu(): void
+ rules(): void
+ displayBoard(): void
+ selectedRegions(): Region
+ advanceTurn(): void
+ game(): void
+ playerInhabits(Region): Player
+ move(): void
+ conflict(): void
+ train(): void
+ build(): void
+ mouseReleased()
+ attackHits(int): int
2...4
1
1
1
18
1
1
1
1
1
1
0...*
*
*
0...*
2
11
Structure
+ c: color
+ location: Region
+ quantity: int
+ display(): void
+ collectRecources(): String
+ getQuantity(): int
+ setQuantity(int): void
+ addStructure(int): void
+ getLocation(): Region
Troop
+ c: color
+ location Region
+ quantity: int
+ Troop(color, Region, int)
+ getLocation(): Region
+ setLocation(Region): void
+ getQuantity(): int
+ setQuantity(int): void
+ addTroops(int): void
+ display(): void
Castle
+ Castle(color, Region, int)
+ display(): void
Village
+ Village(color, Region, int)
+ display(): void
Spinner
+ x: int
+ y: int
+ number: int
+ upDepressed: boolean
+ downDepressed: boolean
+ Spinner(int, int)
+ display(): void
+ upDepressed(): boolean
+ downDepressed(): booleai
+ addNumber(int): void
+ getNumber(): int
+ setNumber(int): void
Button
+ x: int
+ y: int
+ w: int
+ h: int
+ r: int
+ textSize: int
+ c: color
+ pressedColor: color
+ hovorColor: color
+ textColor: color
+ depressed: boolean
+ released: boolean
+ text: String
+ Button(int, int, int, int, int,color, color, color)
+ setText(String,color,int): void
+ display(): void
+ depressed(): boolean
+ released(): boolean
+ setReleased(boolean): void