|  |  |  |
| --- | --- | --- |
| **Concepts used** | **Brief definition/description** | **Give: class/ function/ file name including extension** |
| pointers/smart pointers | * Map\* map in map: point to map class, use it as static * Game \*game in game: point to the game class, use it as static * PlayerStrategy \* stragegy in Player: providing a setStrategy() method to change its own strategy and call relevant that will delegate to a concrete strategy method. * Region\* build() in RegionBuilder: to build a region with its features * Pointer functions in RegionBuilder to set each feature value of a region * Pointer vector in MapRegions and Subject: to use pointer efficiency | **Map.h/.cpp:**  Map\* map;  **Game.h/.cpp:**  Game \* game;  **Player.h/.cpp:**  PlayerStrategy \* stragegy;  void showInfo();  void showInfo(Observer\*);  void showCoins(Observer\* observer);  void showHands(Observer\* observer);  **RegionBuilder.h/.cpp:**  Region\* build();  RegionBuilder\* setId(int id);  RegionBuilder\* setOwner(Owner owner);  RegionBuilder\* setLandform(Landform landform);  RegionBuilder\* setLandMark(LandMark landMark);  RegionBuilder\* setBorder(bool border);  RegionBuilder\* setLostTribes(bool lostTribes);  RegionBuilder\* addGamePieces(GamePieces\* pieces);  **MapRegions.h/.cpp**  vector <RegionBuilder\*> regionBuilder  **subject.h/.cpp**  vector<Observer\*> \*observersList |
| memory management | * Deallocate object in each class * Map Destructor delete 2-D matrix (adjacentcyMatrix) * ~~Player destructor delete its object and strategy pointer~~ | **Map.h/.cpp**  ~Map()  **~~Player.h/.cpp:~~**  ~~~player()~~ |
| vectors | * vector<Region> occupiedRegions: When player occupied some regions, the vector will record the region number * vector<Player> Players: when game starts, select the number of players to player, each player is stored in the vector * vector<vector<int>> numbers: read and store map file and create map * vector <Race> raceVector: randomly choose and store race * vector <Power> powerVector: randomly choose and store power * vector<GamePieces\*> gamePieces: store game pieces (eg: Mountain for Mountain area) * vector<Region\*> regions: build maps * vector <RegionBuilder\*> regionBuilder: store features (eg: boarder, etc) of each region in map * vector<Observer\*> \*observersList: attaching and deattaching observers | **Game.h/.cpp:**  vector<Player> Players  **MapLoader.h/.cpp:**  vector<vector<int> > numbers  **ComboList.h/.cpp:**  vector <Race> raceVector;  vector <Power> powerVector;  **Region.h/.cpp:**  vector<GamePieces\*> gamePieces;  **MapRegions.h/.cpp:**  vector<Region\*> regions;  vector <RegionBuilder\*> regionBuilder;  **Subject.h/.cpp:**  vector<Observer\*> \*observersList |
| data structure | N/A | N/A |
| operator overloading | N/A | N/A |
| file I/O | each map file contains one map information, using file I/O to read the map file and store info into relevant methods | **MapLoader.h/.cpp:**  openFile(string path)  closeInput()  readFile()  fivePlayer.map  fourPlayer.map  threePlayer.map  twoPlayer.map  invaildMap1.map  invaildMap2.map  invaildMap3.map |
| exception handling | * Std::bad\_alloc for operator new and operator new[] when they fail to allocate the requested storage space; * Std::domain\_error for checking whether input value is in domain or not; | **Player.h/.cpp:**  validateInt(string str, int from, int to)  pickRace()  firstConquest ()  reduceTokens()  increaseTokens(int n)  redeployInLoss(int n)  followingConquest()  **Game.h/.cpp:**  getGame()  Initial ()  selectStrategy ()  **Map.h/.cpp:**  getMap() |
| templates | N/A | N/A |
| Any library including GUI | * <stack>: add random number * <vector>: as mentioned above * <String>: using C++ standard [string](https://en.wikipedia.org/wiki/String_(C%2B%2B)) classes and templates. * <iostream>: using input and output fundamentals. * <sstream>: using the template class std::stringstream and other supporting classes for string manipulation. Read map file to the game. * <ctime>: using C-style time/date utilites to generate random nubmer | <stack>  ComboList.h/.cpp  Game.h/.cpp  <string>  Observer  Power and its subclass  Race and its subclass  Region.h/.cpp  Game.h/.cpp  Map.h/.cpp  MapLoader.h/.cpp  <iostream>  DominationView  GamePieces and its subclass  Region.h/.cpp  Game.h/.cpp  Map.h/.cpp  MapLoader.h/.cpp  <sstream>  MapLoader .h/.cpp  <ctime>  Dice.h/.cpp |
| Others | * Using Builder pattern to build game map regions (i.e. id, landform type, landmark type, border, game pieces, etc.) * Using singleton in Game and Map | **MapRegions.h/.cpp:**  createTwoPlayerRegions()  createThreePlayerRegions()  createFourPlayerRegions()  createFivePlayerRegions()  **Map.h/.cpp:**  Map\* map;  Map\* getMap(int)  **Game.h/.cpp:**  Game \* game;  Game \* getGame()  **MapRegions.h/.cpp:**  MapRegions\* mapRegions  Region\* getRegion(int i) |