zMine Preliminary Specification

1. Structural design

For this project we are considering a mining game, where a player has a limited time to amass as much wealth from the earth before the mine collapses. The Main Class creates all the classes as well as starting the Tick method in the world, keeping track of the time. It has 3 ints where you can modify the X/Y of the world length, and the number of players.

First we create a Listener class from the main, basically to listen to keystrokes for player movement from the JFrame. These are sent to the Player class.

Next we initialize our World, which houses everything. We have a xDim, yDim to know the dimensions of the world.

World’s Data Structures

|  |  |
| --- | --- |
| **Data** | **Class** |
| Current players | ArrayList<Player> |
| Current enemies | ArrayList<Enemy> |
| World Tiles | Tile[ ][ ] |

It creates a world made out of tiles for the player to int

1. Object-oriented design

The diagram below shows a class diagram for the Main class.



We have already mentioned *Main*, the class that has *main* and starts the program.

*World* generates the game world with different tiles, spawns players, spawns enemies and establishes a global timer using the tick method. It also enforces collision detection.

*Displayer* draws a visual representation of all game elements like gold count, tiles, players and enemies and updates the visuals systematically.

*Listener* detects keyboard inputs to coordinate movements of players.

*Character* is an abstract class that defines methods used by *Player* and *Enemy*.

* The *Player* extends *Character* to define various attributes of players such as color, position, velocity, gold, bullets, faced-direction and hitpoints.
* The *Enemy* extends *Character* to define various attributes of an enemy such as color, position, velocity, faced-direction, difficulty level and hitpoints.

*Tile* defines the basic attributes of a game tile: color, destroyable and passable.

* *GoldTile* extends *Tile* to define a tile that is unpassable, destroyable and that gives 1000 gold to the player.
* *SilverTile* extends *Tile to* define a tile that is unpassable, destroyable and that gives 500 gold to the player.
* *RegularTile* extends *Tile* to define a default unpassable, destroyable tile that gives no gold when destroyed.
* *TrapTile* extends *Tile* to define a damaging and explosive tile. It kills the player if the player comes within one tile of a *TrapTile*’s location.
* *EntranceTile* extends *Tile* to define a passable and undestroyable spawn tile that marks the player’s spawn point.

1. Detailed design

The detailed specs for the SafeTrade classes have been generated from the Javadoc comments in the source files and are provided in the zMineDocs.zip file in zMine\doc. Open index.html to see the documentation.

1. Testing

There are many different interactions for each individual feature. The enemy class, for example, would need to know if it landed on a trap tile, bumped into a wall, or lost all of its health from a player. Whether or not every variable works as intended will need to be tested and confirmed, otherwise there will be unintended effects, such as the player attacking a wall and not losing health while on a trap tile. Some questions to keep in mind are: Is the enemy allowed to go through walls? Can players go past enemies without losing health? What if the tiles don’t work as intended?

Enemy and player will need a subclass called character as a base model for methods and constructors. A subclass will also need to be created for different type of tiles, such as silver tile, gold tile, and trap. Test the world class to see if the world is made to desired specifications. Find out if every class are working as intended in the world class.