vim-markdown-preview.html

DESIGN

Entity

Entities are an object that exists in the WorldModel.

```
+ getAnimationPeriod() : int
+ nextImage()
+ getCurrentTile() : Tile
+ transformNotFull(WorldModel, EventSchedule)
+ transformFull(WorldModel, EventSchedule)
+ moveToFull(WorldModel, Entity, EventSchedule) : boolean
+ moveToNotFull(WorldModel, Entity, EventSchedule) : boolean
```

All of these methods act on a specific instace of an Entity object and cannot exist without it. In the case of getCurrentTile, it returns the tile of the current instance of the class; in the case of the move* methods, it changes the state of one individual object, which is why they are instance methods.

```
+ static scheduleActions(Entity, EventSchedule, WorldModel)
```

This is one I wasn't sure of. The schedule action method seems like something that would be indipendent of any object, including where Entity is null, leading it to be static. I also wasn't sure weather to make it under the entity class or not, but since it schedules actions that are associated with Enitities, I chose this class.

```
+ moveToOreBlob(WorldModel, Entity, EventSchedule) : boolean
- nextPositionMiner(WorldModel, Point) : Point
- nextPositionOreBlob(WorldModel, Point) : Point
```

Again, these are acting on one entity object, therefore they should be instance methods of the Enitity class. Since next* are only called from within the class, they are private.

```
+ static createBlacksmith(Point) : Entity
- static createMinerFull(int, Point, int, int) : Entity
+ static createMinerNotFull(int, Point, int, int) : Entity
+ static createObstacle(Point) : Entity
+ static createOre(Point, int) : Entity
+ static createOreBlob(Point, int, int) : Entity
+ static createQuake(Point) : Entity
+ static createVein(Point, int) : Entity
```

All of these are specified by the docs, they are static because they are not connected to a specific instance of Entity, and public because they can be created anywhere.

Action

An action is a the abstract of something that will happen during the playtime.

```
+ executeAction(EventSchedule)
+ executeAnimationAction(EventSchedule)
```

```
+ executeActivityAction(EventSchedule)
```

These three were fairly ambiguous as well, but I put these as instance methods in Action because they all execute actions with respect to the EventSchedule, thus they should be part of action.

```
- executeMinerFullActivity(WorldModel, EventSchedule)
- executeMinerNotFullActivity(WorldModel, EventSchedule)
- executeOreActivity(WorldModel, EventSchedule)
- executeOreBlobActivity(WorldModel, EventSchedule)
- executeQuakeActivity(WorldModel, EventSchedule)
- executeVeinActivity(WorldModel, EventSchedule)
```

All of these are executing an action specific to each Enitity, they are also currently only used in this class, therefore they are private.

VirtualWorld

```
+ static paint(WorldModel, AnimationFrame)
```

Since this method only interacts with the view, the appearence of the class, it is in virtual world.

WorldModel

```
+ getSize() : Size
+ getBackground() : Tile
+ getOccupant() : Entity
+ getEntities() Set<Entity>
+ setBackground(x,y,Tile)
+ getOccupant(Point) : Entity
+ getOccupantCell(Point) : Entity
+ setOccupantCell(Point, Entity)
+ removeEntity(Entity)
- removeEntityAt(Point)
```

Most of these are getters and setters, the rest are calling for information about points or setting that information. They are all only interacting with the worldmodel so they should be in that class.

```
+ addEntity(Entity)
+ isOccupied(Point) : boolean
+ withinBounds(Point)
+ findOpenAround(Point) : Point
+ moveEntity(Entity, Point)
+ findNearest(WorldModel, Point, EntityKind) : Entity
- nearestEntity(List[Entity], Point) : Entity
+ createActivityAction(Entity) : Action
```

All of these are are editing or getting data from the world model, they are effectively getters and setters. Nearest entity is private because it is only referenced within the class. I have no idea where createActivityAction should really go, but it fits in with the rest of the methods here, so here it will stay.

```
+ static createAnimationAction(Entity, int) : Action
```

This is static because the animation actions are indipendent of the WM, but fit in with the createActivityAction.