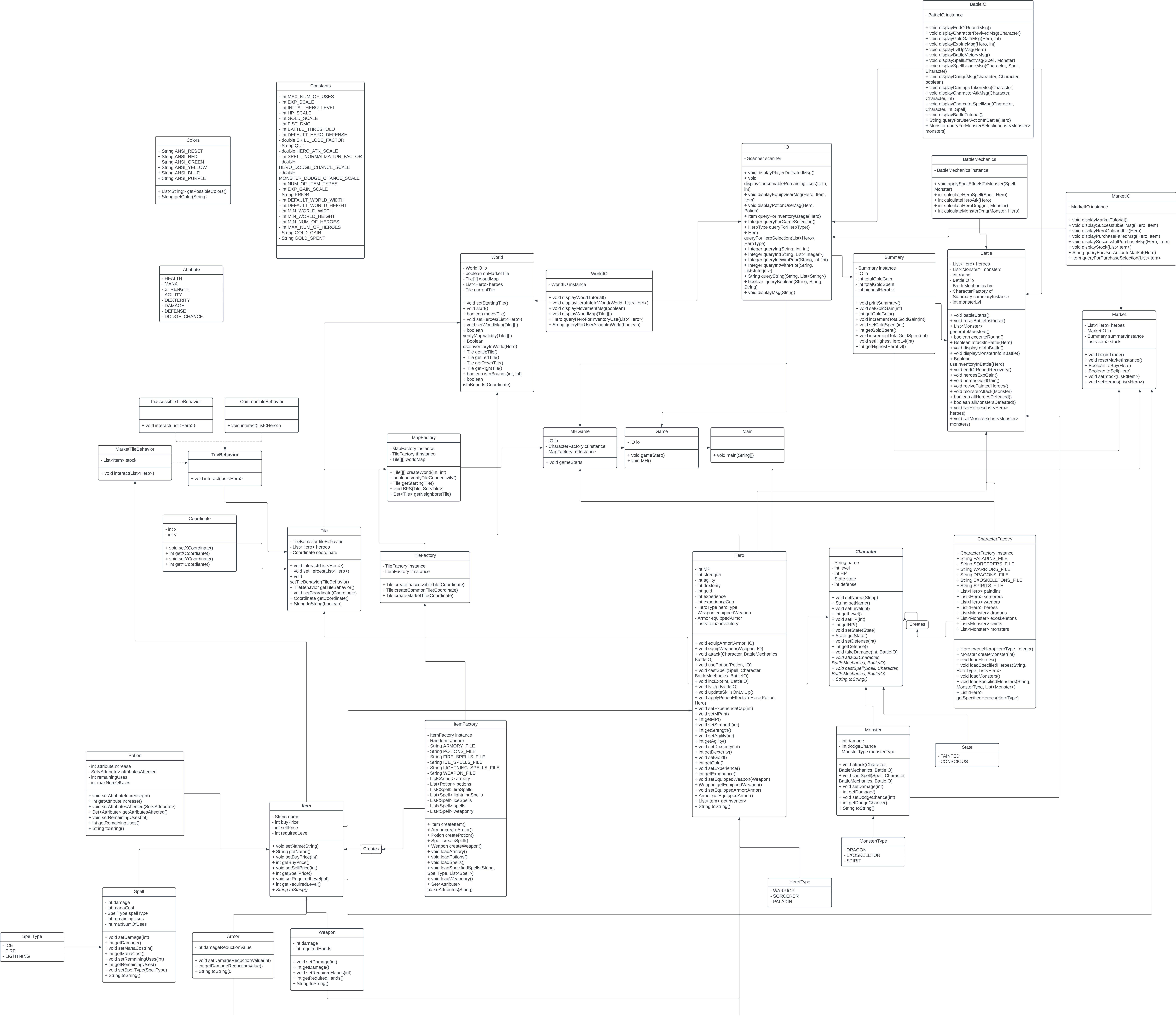

```
BattleIO
- BattleIO instance
+ void displayEndOfRoundMsg()
+ void displayCharacterRevivedMsg(Character)
+ void displayGoldGainMsg(Hero, int)
+ void displayExpIncMsg(Hero, int)
+ void displayLvlUpMsg(Hero)
+ void displayBattleVictoryMsg()
+ void displaySpellEffectMsg(Spell, Monster)
+ void displaySpellUsageMsg(Character, Spell,
Character)
+ void displayDodgeMsg(Character, Character,
boolean)
+ void displayDamageTakenMsg(Character)
+ void displayCharacterAtkMsg(Character,
Character, int)
+ void displayCharcaterSpellMsg(Character,
Character, int, Spell)
+ void displayBattleTutorial()
+ String queryForUserActionInBattle(Hero)
+ Monster queryForMonsterSelection(List<Monster>
monsters)
```

```
Colors
+ String ANSI_RESET
+ String ANSI_RED
+ String ANSI_GREEN
+ String ANSI_YELLOW
+ String ANSI_BLUE
+ String ANSI_PURPLE
+ List<String> getPossibleColors()
+ String getColor(String)
```

```
Constants
- int MAX_NUM_OF_USES
- int EXP_SCALE
- int INITIAL_HERO_LEVEL
- int HP_SCALE
- int GOLD_SCALE
- int FIST_DMG
- int BATTLE_THRESHOLD
- int DEFAULT_HERO_DEFENSE
- double SKILL_LOSS_FACTOR
- String QUIT
- double HERO_ATK_SCALE
- int SPELL_NORMALIZATION_FACTOR
- double
HERO_DODGE_CHANCE_SCALE
- double
MONSTER_DODGE_CHANCE_SCALE
- int NUM_OF_ITEM_TYPES
- int EXP_GAIN_SCALE
- String PRIOR
- int DEFAULT_WORLD_WIDTH
- int DEFAULT_WORLD_HEIGHT
- int MIN_WORLD_WIDTH
- int MIN_WORLD_HEIGHT
- int MIN_NUM_OF_HEROES
- int MAX_NUM_OF_HEROES
- String GOLD_GAIN
- String GOLD_SPENT
```

```
Attribute
- HEALTH
- MANA
- STRENGTH
- AGILITY
- DEXTERITY
- DAMAGE
- DEFENSE
- DODGE_CHANCE
```

```
IO
- Scanner scanner
+ void displayPlayerDefeatedMsg()
+ void
displayConsumableRemainingUses(Item,
int)
+ void displayEquipGearMsg(Hero, Item,
Item)
+ void displayPotionUseMsg(Hero,
Potion)
+ Item queryForInventoryUsage(Hero)
+ Integer queryForGameSelection()
+ HeroType queryForHeroType()
+ Hero
queryForHeroSelection(List<Hero>,
HeroType)
+ Integer queryInt(String, int, int)
+ Integer queryInt(String, List<Integer>)
+ Integer queryIntWithPrior(String, int, int)
+ Integer queryIntWithPrior(String,
List<Integer>)
+ String queryString(String, List<String>)
+ boolean queryBoolean(String, String,
String)
+ void displayMsg(String)
```

```
BattleMechanics
- BattleMechanics instance
+ void applySpellEffectsToMonster(Spell,
Monster)
+ int calculateHeroSpell(Spell, Hero)
+ int calculateHeroAtk(Hero)
+ int calculateHeroDmg(int, Monster)
+ int calculateMonsterDmg(Monster, Hero)
```

```
MarketIO
- MarketIO instance
+ void displayMarketTutorial()
+ void displaySuccessfulSellMsg(Hero, Item)
+ void displayHeroGoldandLvl(Hero)
+ void displayPurchaseFailedMsg(Hero, Item)
+ void displaySuccessfulPurchaseMsg(Hero, Item)
+ void displayStock(List<Item>)
+ String queryForUserActionInMarket(Hero)
+ Item queryForPurchaseSelection(List<Item>
```

```
World
- WorldIO io
- boolean onMarketTile
- Tile[][] worldMap
- List<Hero> heroes
- Tile currentTile
+ void setStartingTile()
+ void start()
+ boolean move(Tile)
+ void setHeroes(List<Hero>)
+ void setWorldMap(Tile[][])
+ boolean
verifyMapValidity(Tile[][])
+ Boolean
useInventoryInWorld(Hero)
+ Tile getUpTile()
+ Tile getLeftTile()
+ Tile getDownTile()
+ Tile getRightTile()
+ boolean isInBounds(int, int)
+ boolean
isInBounds(Coordinate)
```

```
WorldIO
- WorldIO instance
+ void displayWorldTutorial()
+ void displayHeroInfoInWorld(World, List<Hero>)
+ void displayMovementMsg(boolean)
+ void displayWorldMap(Tile[][])
+ Hero queryHeroForInventoryUse(List<Hero>)
+ String queryForUserActionInWorld(boolean)
```

```
Summary
- Summary instance
- IO io
- int totalGoldGain
- int totalGoldSpent
- int highestHeroLvl
+ void printSummary()
+ void setGoldGain(int)
+ int getGoldGain()
+ void incrementTotalGoldGain(int)
+ void setGoldSpent(int)
+ int getGoldSpent()
+ void incrementTotalGoldSpent(int)
+ void setHighestHeroLvl(int)
+ int getHighestHeroLvl()
```

```
Battle
- List<Hero> heroes
- List<Monster> monsters
- int round
- BattleIO io
- BattleMechanics bm
- CharacterFactory cf
- Summary summaryInstance
- int monsterLvl
+ void battleStarts()
+ void resetBattleInstance()
+ List<Monster>
generateMonsters()
+ boolean executeRound()
+ Boolean attackInBattle(Hero)
+ void displayInfoInBattle()
+ void displayMonsterInfoInBattle()
+ Boolean
useInventoryInBattle(Hero)
+ void endOfRoundRecovery()
+ void heroesExpGain()
+ void heroesGoldGain()
+ void reviveFaintedHeroes()
+ void monsterAttack(Monster)
+ boolean allHeroesDefeated()
+ boolean allMonstersDefeated()
+ void setHeroes(List<Hero>
heroes)
+ void setMonsters(List<Monster>
monsters)
```

```
Market
- List<Hero> heroes
- MarketIO io
- Summary summaryInstance
- List<Item> stock
+ void beginTrade()
+ void resetMarketInstance()
+ Boolean toBuy(Hero)
+ Boolean toSell(Hero)
+ void setStock(List<Item>)
+ void setHeroes(List<Hero>)
```

```
InaccessibleTileBehavior
+ void interact(List<Hero>)
```

```
CommonTileBehavior
+ void interact(List<Hero>)
```

```
MarketTileBehavior
- List<Item> stock
+ void interact(List<Hero>)
```

```
TileBehavior
+ void interact(List<Hero>
```

```
MapFactory
- MapFactory instance
- TileFactory tfInstance
- Tile[][] worldMap
+ Tile[][] createWorld(int, int)
+ boolean verifyTileConnectivity()
+ Tile getStartingTile()
+ void BFS(Tile, Set<Tile>)
+ Set<Tile> getNeighbors(Tile)
```

```
MHGame
- IO io
- CharacterFactory cfInstance
- MapFactory mfInstance
+ void gameStarts
```

```
Game
- IO io
+ void gameStart()
+ void MH()
```

```
Main
+ void main(String[])
```

```
Coordinate
- int x
- int y
+ void setXCoordinate()
+ int getXCoordiante()
+ void setYCoordinate()
+ int getYCoordiante()
```

```
Tile
- TileBehavior tileBehavior
- List<Hero> heroes
- Coordinate coordinate
+ void interact(List<Hero>)
+ void setHeroes(List<Hero>)
+ void
setTileBehavior(TileBehavior)
+ TileBehavior getTileBehavior()
+ void setCoordinate(Coordinate)
+ Coordinate getCoordinate()
+ String toString(boolean)
```

```
TileFactory
- TileFactory instance
- ItemFactory ifInstance
+ Tile createInaccessibleTile(Coordinate)
+ Tile createCommonTile(Coordinate)
+ Tile createMarketTile(Coordinate)
```

```
Hero
- int MP
- int strengtth
- int agility
- int dexterity
- int gold
- int experience
- int experienceCap
- HeroType heroType
- Weapon equippedWeapon
- Armor equippedArmor
- List<Item> inventory
+ void equipArmor(Armor, IO)
+ void equipWeapon(Weapon, IO)
+ void attack(Character, BattleMechanics,
BattleIO)
+ void usePotion(Potion, IO)
+ void castSpell(Spell, Character,
BattleMechanics, BattleIO)
+ void incExp(int, BattleIO)
+ void lvlUp(BattleIO)
+ void updateSkillsOnLvlUp()
+ void applyPotionEffectsToHero(Potion,
Hero)
+ void setExperienceCap(int)
+ void setMP(int)
+ int getMP()
+ void setStrength(int)
+ int getStrength()
+ void setAgility(int)
+ int getAgility()
+ void setDexterity(int)
+ int getDexterity()
+ void setGold()
+ int getGold()
+ void setExperience()
+ int getExperience()
+ void setEquippedWeapon(Weapon)
+ Weapon getEquippedWeapon()
+ void setEquippedArmor(Armor)
+ Armor getEquippedArmor()
+ List<Item> getInventory
+ String toString()
```

```
Character
- String name
- int level
- int HP
- State state
- int defense
+ void setName(String)
+ String getName()
+ void setLevel(int)
+ int getLevel()
+ void setHP(int)
+ int getHP()
+ void setState(State)
+ State getState()
+ void setDefense(int)
+ int getDefense()
+ void takeDamage(int, BattleIO)
+ void attack(Character,
BattleMechanics, BattleIO)
+ void castSpell(Spell, Character,
BattleMechanics, BattleIO)
+ String toString()
```

```
CharacterFacotry
+ CharacterFactory instance
+ String PALADINS_FILE
+ String SORCERERS_FILE
+ String WARRIORS_FILE
+ String DRAGONS_FILE
+ String EXOSKELETONS_FILE
+ String SPIRITS_FILE
+ List<Hero> paladins
+ List<Hero> sorcerers
+ List<Hero> warriors
+ List<Hero> heroes
+ List<Monster> dragons
+ List<Monster> exoskeletons
+ List<Monster> spirits
+ List<Monster> monsters
+ Hero createHero(HeroType, Integer)
+ Monster createMonster(int)
+ void loadHeroes()
+ void loadSpecifiedHeroes(String,
HeroType, List<Hero>
+ void loadMonsters()
+ void loadSpecifiedMonsters(String,
MonsterType, List<Monster>)
+ List<Hero>
getSpecifiedHeroes(HeroType)
```

```
Monster
- int damage
- int dodgeChance
- MonsterType monsterType
+ void attack(Character,
BattleMechanics, BattleIO)
+ void castSpell(Spell, Character,
BattleMechanics, BattleIO)
+ void setDamage(int)
+ int getDamage()
+ void setDodgeChance(int)
+ int getDodgeChance()
+ String toString()
```

```
State
- FAINTED
- CONSCIOUS
```

```
MonstertType
- DRAGON
- EXOSKELETON
- SPIRIT
```

```
HerotType
- WARRIOR
- SORCERER
- PALADIN
```

```
Potion
- int attributeIncrease
- Set<Attribute> attributesAffected
- int remainingUses
- int maxNumOfUses
+ void setAttributeIncrease(int)
+ int getAttributeIncrease()
+ void setAttributesAffected(Set<Attribute>)
+ Set<Attribute> getAttributesAffected()
+ void setRemainingUses(int)
+ int getRemainingUses()
+ String toString()
```

```
Item
- String name
- int buyPrice
- int sellPrice
- int requiredLevel
+ void setName(String)
+ String getName()
+ void setBuyPrice(int)
+ int getBuyPrice()
+ void setSellPrice(int)
+ int getSpellPrice()
+ void setRequiredLevel(int)
+ int getRequiredLevel()
+ String toString()
```

```
ItemFactory
- ItemFactory instance
- Random random
- String ARMORY_FILE
- String POTIONS_FILE
- String FIRE_SPELLS_FILE
- String ICE_SPELLS_FILE
- String LIGHTNING_SPELLS_FILE
- String WEAPON_FILE
- List<Armor> armory
- List<Potion> potions
- List<Spell> fireSpells
- List<Spell> lightningSpells
- List<Spell> iceSpells
- List<Spell> spells
- List<Spell> weaponry
+ Item createItem()
+ Armor createArmor()
+ Potion createPotion()
+ Spell createSpell()
+ Weapon createWeapon()
+ void loadArmory()
+ void loadPotions()
+ void loadSpells()
+ void loadSpecifiedSpells(String,
SpellType, List<Spell>)
+ void loadWeaponry()
+ Set<Attribute>
parseAttributes(String)
```

```
SpellType
- ICE
- FIRE
- LIGHTNING
```

```
Spell
- int damage
- int manaCost
- SpellType spellType
- int remainingUses
- int maxNumOfUses
+ void setDamage(int)
+ int getDamage()
+ void setManaCost(int)
+ int getManaCost()
+ void setRemainingUses(int)
+ int getRemainingUses()
+ void setSpellType(SpellType)
+ String toString()
```

```
Armor
- int damageReductionValue
+ void setDamageReductionValue(int)
+ int getDamageReductionValue()
+ String toString(0
```

```
Weapon
- int damage
- int requiredHands
+ void setDamage(int)
+ int getDamage()
+ void setRequiredHands(int)
+ int getRequiredHands()
+ String toString()
```

Our class structure, as you can see, can be generally classified into a few chunks where their related functionalities are pretty enclosed. The chunks are Item, Character, Tile/Map, IO/Worlds. The program starts from a manager class called Game, where it fires up the corresponding game depending on the user choice. This wrapper class allows the instantiating of one game to be separated from the other, aligning with the single responsibility principle. The use of Factory Pattern to create Tile, Item and Character isolates the logic and responsibility of object creation within their respective factories. The decision to have multiple subclasses of IO is to purposely house different displays and queries that are fit only under specific situations. The choice to have a Constant class allows us to house general constants that we may often use throughout the program, and thus void redundancy. Furthermore, the choice of using enums instead of declaring them as "static final constants" gives us a type that is not String, but a type that is more related to the task each enum is responsible for.

Again, the use of Factory Pattern hides the logic of object creation away from classes that needs these objects, this does not only allow for easier modification of object creation in the future as all the logics are in one place, but also strictly aligns with the single responsibility principle as another class would no longer be responsible for the creation of any object. This would also allow for the extension to create different heroes/tiles/items in the future by only needing to make changes to their respective factory.

Some code that I used from past assignments is the Colors.java, borrowed from Michelle with her permission to use it, and it is for aesthetic and user clarity reasons, to give certain texts more highlights. For future assignment where we may need to implement tiles of different behaviors, I am thinking about continuing to extend from the TileBehavior interface, via the Strategy Pattern, to achieve the different behaviors needed