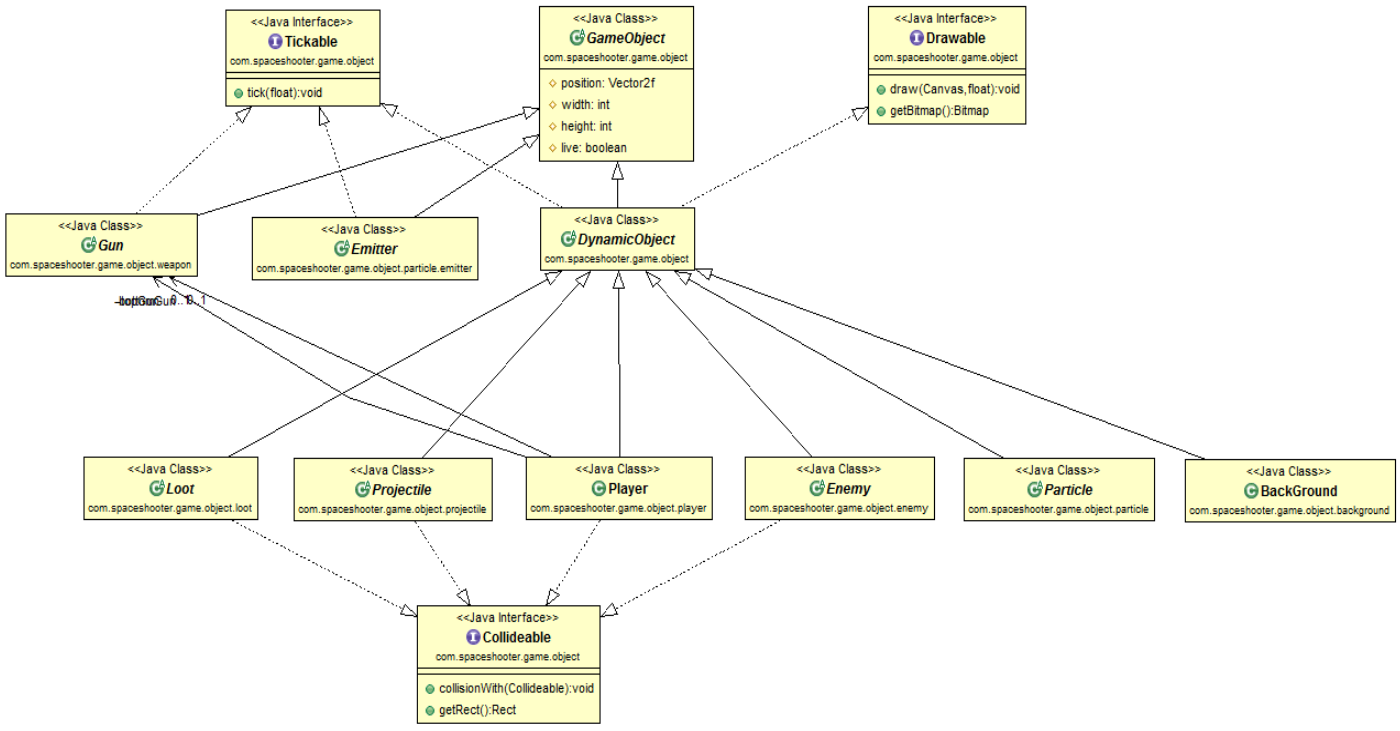
UML

This is an uml diagram of our gameobject system. At the top of the “tree” we have the abstract class GameObject then we have DynamicObject that inherits from GameObject and implements the interfaces Tickable and Drawable. Then you can see that most of our gameobjects are dynamicobjects which means they should be updated and drawn. Starting from the left we have the abstract class Loot which represents the loot that gets dropped from enemies in the game, for instance we have healthpacks and slowtime power up that both are children of the Loot class. Then we have the abstract class Projectile which is used for representing the enemy and player projectiles in the game. Here you can see that they implement the Collideable interface which means that they should get checked for collisions. We then come to the Player class which simply represents the Player and it should of course be checked for collisions. The Enemy class simply represents all the enemies in the game and should also be checked for collisions. The last two classes are the particle and background. Particles are short lived small dynamicobjects which we use to simulate explosions and the engine on the ship. The background is also only a dynamicobject since it should get scrolled.

We then have two types of gameobjects that should only get updated and not drawn: The guns and particle emitters. So when creating them you simply inherit the GameObject class and implement the tickable interface. The Gun class represents the different guns the player can equip on the ship and the particle emitter is used for generating particle effects. Lastly we can see that the player class has two guns: a bottom one and a top one.


<<Java Interface>>
Tickable
com.spaceshooter.game.object
tick(float):void
<<Java Class>>
GameObject
com.spaceshooter.game.object
position: Vector2f
width: int
height: int
live: boolean
<<Java Interface>>
Drawable
com.spaceshooter.game.object
draw(Canvas,float):void
getBitmap():Bitmap
<<Java Class>>
Gun
com.spaceshooter.game.object.weapon
<<Java Class>>
Emitter
com.spaceshooter.game.object.particle.emitter
<<Java Class>>
DynamicObject
com.spaceshooter.game.object
-bottomGun 0..1
-topGun 0..1
<<Java Class>>
Loot
com.spaceshooter.game.object.loot
<<Java Class>>
Projectile
com.spaceshooter.game.object.projectile
<<Java Class>>
Player
com.spaceshooter.game.object.player
<<Java Class>>
Enemy
com.spaceshooter.game.object.enemy
<<Java Class>>
Particle
com.spaceshooter.game.object.particle
<<Java Class>>
BackGround
com.spaceshooter.game.object.background
<<Java Interface>>
Collideable
com.spaceshooter.game.object
collisionWith(Collideable):void
getRect():Rect