# Refactoring a Real Time STRATEGY game

Grunt

Tank

The GRUNTS IN SPACE ©: RTS game has two different units that have different behavior for how they move and attack. See the four class files below:

```
public class TestRTS {
        public static void main(String[] args) {
                GameUnit grunt = new Grunt();
                GameUnit tank = new Tank();
                grunt.walk();
                tank.walk();
                grunt.attack();
                tank.attack();
        }
}
```

```
public class Grunt extends GameUnit {

  public Grunt(){
   System.out.println("new Grunt");
  }

 public void attack(){
   System.out.println("Grunt:hacking with axe");
  }
}
```

```
public class GameUnit {

  public void attack(){}

  public void walk() {
        System.out.println(this.getClass().getName()
        +":walking");
  }
}
```

```
    public class Tank extends GameUnit {
      public Tank(){
            System.out.println("new Tank");
      }

      public void attack(){
            System.out.println("Tank:shooting with canon");
      }
      public void walk(){
            //overide tanks don't walk
            System.out.println("Tank:driving");
            }
    }
```

In order to make the game more interesting to play, the game designer of this game decides that units can pick up different weapons and upgrades that change the walking behavior and attack behaviour for each unit.

| | Grunt | Tank |
|---|---|---|
| moving | {walking} | {driving, flying} |
| attacking | {Axe, Pistol} | {Canon, Rocket} |

Assignment 1: Can you identify some limitations with the current class design? Does this design facilitate change?

Assignment 2: Refactor these classes using the Strategy pattern. Identify those parts that change and "factor" these out. Use the program to an interface not to an implementation paradigm. You probably also want to rename walk to move (initially the game only had walking characters). Each unit at any time only has one type of walking and attacking behavior.

Assignment 3: make it so that they can change their behavior when they pick up a new weapon or find an upgrade.

Assignment 4: Try to draw a class diagram for your solution using TinyUML

Assignment 5: For each one of the three Object Oriented Design Principles, (inheritance, encapsulation, polymorphism) indicate briefly one instance of where it has been implemented in your application

Assignment 6: Can you name a disadvantage of implementing the Strategy pattern?