# 1. anyMatch по exisitng instance на обект.

Ако се съдържа обект с подадения в параметъра клас (Component component), ще хвърлим exception.

В противен случай добавяме компонента.

@Override  
public void addComponent(Component component) {  
 if (components.stream().anyMatch(c -> c.getClass() == component.getClass())) {

throw new IllegalArgumentException(String.format(EXISTING\_COMPONENT,  
 component.getClass().getSimpleName(),  
 this.getClass().getSimpleName(), this.getId()));  
 }  
 this.components.add(component);  
}

# 2. noneMatch + remove By Index

Ако не намерим == noneMatch връща true и влизаме в проверката.

@Override  
public Component removeComponent(String componentType) {

if (components.isEmpty() || components.stream().noneMatch(c -> c.getClass().getSimpleName().equals(componentType))) {

throw new IllegalArgumentException(String.format(NOT\_EXISTING\_COMPONENT, componentType,  
 this.getClass().getSimpleName(), this.getId()));  
 }  
  
 int index = 0;  
 for (int i = 0; i < components.size(); i++) {  
 Component component = components.get(i);  
 if (component.getClass().getSimpleName().equals(componentType)) {  
 index = i;  
 break;  
 }  
 }  
 return components.remove(index);  
}

# 3. Instance of + anyMatch

@Override  
public String start(Collection<Player> players) {  
 Collection<Player> terrorists = players  
 .stream()  
 .filter(p -> p instanceof Terrorist)  
 .collect(Collectors.*toList*());  
  
 Collection<Player> counterTerrorists = players  
 .stream()  
 .filter(p -> p instanceof CounterTerrorist)  
 .collect(Collectors.*toList*());  
  
 while (counterTerrorists.stream().anyMatch(Player::isAlive) && terrorists.stream().anyMatch(Player::isAlive)) {  
  
 for (Player terrorist : terrorists) {  
 for (Player counterTerrorist : counterTerrorists) {  
 counterTerrorist.takeDamage(terrorist.getGun().fire());  
 }  
 }  
 counterTerrorists = counterTerrorists.stream().filter(Player::isAlive).collect(Collectors.*toList*());  
  
 for (Player counterTerrorist : counterTerrorists) {  
 for (Player terrorist : terrorists) {  
 terrorist.takeDamage(counterTerrorist.getGun().fire());  
 }  
 }  
 terrorists = terrorists.stream().filter(Player::isAlive).collect(Collectors.*toList*());  
 }  
  
 for (Player terrorist : terrorists) {  
 if (terrorist.isAlive()) {  
 return *TERRORIST\_WINS*;  
 }  
 }  
 return *COUNTER\_TERRORIST\_WINS*;  
  
}

# 4. Repository

package CounterStriker.repositories;  
  
import CounterStriker.models.guns.Gun;  
  
import java.util.ArrayList;  
import java.util.Collection;  
  
import static CounterStriker.common.ExceptionMessages.*INVALID\_GUN\_REPOSITORY*;  
import static CounterStriker.common.ExceptionMessages.*INVALID\_PLAYER\_REPOSITORY*;  
  
public class GunRepository<T extends Gun> implements Repository<T> {  
 Collection<T> models;  
  
 public GunRepository() {  
 this.models = new ArrayList<>();  
 }  
  
 @Override  
 public Collection<T> getModels() {  
 return this.models;  
 }  
  
 @Override  
 public void add(T model) {  
 if (model == null) {  
 throw new NullPointerException(*INVALID\_PLAYER\_REPOSITORY*);  
 }  
 models.add(model);  
 }  
  
 @Override  
 public boolean remove(T model) {  
 return this.models.remove(model);  
 }  
  
 @Override  
 public T findByName(String name) {  
 return this.models.stream().filter(m -> m.getName().equals(name)).findFirst().orElse(null);  
 }  
}

# 5. Find First

@Override  
 public T findByName(String name) {  
 return this.models.stream().filter(m -> m.getName().equals(name)).findFirst().orElse(null);  
 }  
}

6. Sort

Returns information about each player separated with a new line. Order them by type alphabetically, then by health descending, then by username alphabetically. You can use the overridden **.toString() Player** method.

**"{player type}: {player username}**

**--Health: {player health}**

**--Armor: {player armor}**

**--Gun: {player gun name}"**

@Override  
 public String report() {  
 StringBuilder sb = new StringBuilder();  
  
 this.players.getModels().stream().sorted((p1, p2) -> {  
 int result = p1.getClass().getSimpleName().compareTo(p2.getClass().getSimpleName());  
 if (result == 0) {  
 result = Integer.*compare*(p2.getHealth(), p1.getHealth());  
 }  
 if (result == 0) {  
 result = p1.getUsername().compareTo(p2.getUsername());  
 }  
 return result;  
 }).forEach(p -> sb.append(p.toString()).append(System.*lineSeparator*()));  
  
 return sb.toString().trim();  
 }  
}