**Implementatietemplate Code Smell – Shothun Surgery Code Smell**

## Aanpak

Geef aan jullie aanpak was bij het opsporen en oplossen van de shotgun surgery code smell (1-5 zinnen):

|  |
| --- |
| Onze scrumspel heeft hele lange methodes. Als we iets eraan veranderen aan de methode, moeten we gelijk ook meerdere dingen aanpassen. Dit kan veel tijd en energie kosten. We hebben een kamerFactory gemaakt zodat alle kamers daarin worden gemaakt. We hebben ook een CommandHandler gemaakt waar al onze commands zullen zijn. Door deze veranderingen kunnen we gemakkelijk code aanpassen zonder onze code te hoeven breken. |

## Code smells:

|  |  |
| --- | --- |
| **Shotgun**  **Surgery**  **Code Smell**  **1** | Plak hier de code van 2 methodes waar de shotgun surgery code smell 1 was geimplementeerd |
| private void initializeKamers() {  kamers.put(1, SprintPlanningKamer.*maakKamer*(speler.getInventory()));  kamers.put(2, DailyScrumKamer.*maakKamer*(speler.getInventory()));  kamers.put(3, ScrumBoardKamer.*maakKamer*(speler.getInventory()));  kamers.put(4, SprintReviewKamer.*maakKamer*(speler.getInventory()));  kamers.put(5, SprintRetrospectiveKamer.*maakKamer*(speler.getInventory()));  kamers.put(6, FinaleTIAKamer.*maakKamer*(speler.getInventory())); } |
| Plak hier de code die de shotgun surgery code smell 1 oplost |
| public void startSpel() {  System.*out*.println("Voer je naam in: ");  String naam = scanner.nextLine();   speler = new Speler(naam, 100);  kamers = KamerFactory.*maakAlleKamers*(speler.getInventory()); speler.setHuidigeKamer(1);  SQLLoader.*loadFromDatabase*("speler");   roomChanger = new RoomChanger(speler, kamers);   // Monster is now an observer, needs speler reference  monster = new Monster(40, speler);   // Register observers for all rooms  for (Kamer kamer : kamers.values()) {  // Add both door and monster as observers  kamer.addObserver(kamer.getDeur(), monster);  }    System.*out*.println("\nWelkom bij het ScrumSpel, " + naam + "!");  System.*out*.println("Je begint in kamer " + speler.getHuidigeKamer() + " met " + speler.getHp() + " HP.");   hoofdSpelLoop();  scanner.close(); } |
| **Shotgun**  **Surgery**  **Code Smell**  **2** | Plak hier de code van 2 methodes waar de shotgun surgery code smell 2 was geimplementeerd |
| private void hoofdSpelLoop() {  boolean spelActief = true;   while (spelActief) {  if (speler.getHp() <= 0) {  handleGameOver();  continue;  }   roomChanger.toonHuidigeKamer();   System.*out*.print("\n> ");  String input = scanner.nextLine().toLowerCase().trim();   switch (input) {  case "status":  speler.toonStatus();  break;  case "inventory":  speler.getInventory().toonInventory();  break;  case "info":  getHuidigeKamer().toonKamerinfo();  break;  case "stop":  System.*out*.println("Spel opslaan en afsluiten...");  SQLSaver.*saveToDatabase*(speler);  spelActief = false;  break;   case "joker":  Joker gekozenJoker = speler.kiesJoker();  if (gekozenJoker instanceof Item) {  getHuidigeKamer().accepteer(gekozenJoker);  speler.getInventory().verwijderItem(((Item) gekozenJoker).getNaam());  }  break;   case "beantwoord":  boolean correct = getHuidigeKamer().handlePlayerAnswer();  if (correct) {  SQLSaver.*saveToDatabase*(speler);  }  break;  default:  if (input.startsWith("ga naar kamer ")) {  roomChanger.verwerkKamerVerandering(input);  } else {  System.*out*.println("Onbekend commando. Beschikbare commando's:");  System.*out*.println("- 'beantwoord' : Probeer de vraag te beantwoorden");  System.*out*.println("- 'ga naar kamer X' : Ga naar kamer X (als unlocked)");  System.*out*.println("- 'status' : Toon je huidige status");  System.*out*.println("- 'stop' : Sla op en stop het spel");  }  }  }  scanner.close(); } |
| Plak hier de code die de shotgun surgery code smell 2 oplost |
| private void hoofdSpelLoop() {  spelActief = true;  while (spelActief) {  if (speler.getHp() <= 0) {  handleGameOver();  continue;  }   roomChanger.toonHuidigeKamer();   System.*out*.print("\n> ");  String input = scanner.nextLine().toLowerCase().trim();   CommandHandler commandHandler = new CommandHandler(speler, getHuidigeKamer(), scanner, () -> spelActief = false);     if (!commandHandler.verwerkInput(input)) {  if (input.startsWith("ga naar kamer ")) {  roomChanger.verwerkKamerVerandering(input);  } else {  System.*out*.println("Onbekend commando. Beschikbare commando's:");  System.*out*.println("- 'beantwoord' : Probeer de vraag te beantwoorden");  System.*out*.println("- 'ga naar kamer X' : Ga naar kamer X (als unlocked)");  System.*out*.println("- 'status' : Toon je huidige status");  System.*out*.println("- 'stop' : Sla op en stop het spel");  }  }    }  scanner.close(); } |
| **Shotgun**  **Surgery**  **Code Smell**  **3** | Plak hier de code van 2 methodes waar de shotgun surgery code smell 3 was geimplementeerd |
|  |
| Plak hier de code die de shotgun surgery code smell 3 oplost |
|  |

**Bijdrage per student. Geef aan per student wat zijn of haar bijdrage was.**

|  |  |
| --- | --- |
| Naam student | Bijdrage |
| Dami Kastaneer | Kamer Factory gemaakt en geimplementeerd. Command interface gemaakt en geimplementeerd |
| Anson Mok | Code getest |
| Nathaniel Rasmijn | Code getest |
|  |  |