David Joan – 2501989784

Berikut penjelasan dari smell code yang saya sudah fix.

Penjelasan terdiri dari : Penjelasan singkat, code before, code after

NOTE: code dalam word akan berupa table, jika ingin melihat code secara keseluruhan (before & after), dapat di lihat pada link git hub berikut, <https://github.com/Zappie733/UTS-Code-Reengineering.git>

|  |
| --- |
| SMELL 5, Bloaters(Long Method) dan dispensables(comments) - > extract method  Pada method actionPerformed dapat dilihat ada beberapa logic yaitu untuk ngecheck food, check collision, dan move. Code dari logic2 tersebut agak sulit untuk dimengerti oleh karena itu dibantu dengan adanya inline comment (comments smell). Selain dari adanya comments smell dapat dilihat juga bahwa method actionPerformed ini memiliki responsibilitas lebih dari 1 dapat diliht dari adanya variasi logic untuk berbagai hal (long method smell).  Untuk menghi;angkan smell comments dan juga long method, saya melakukan extract method dengan penamaan yang dapat mendeskripsikan isi method. Jadi ketiga logic yang ada saya pisah jadi 3 method yang berbeda dan diberikan nama yang sesuai. |
| BEFORE |
| @Override      public void actionPerformed(ActionEvent e) {          if (alive) {              // check food              if ((pos[0].x == food\_x) && (pos[0].y == food\_y)) {                  length++;                  randomFood();                  int d = DELAY-length;                  if(d < 40) d = 40;                  timer.setDelay(d);              }              // checkCollision              for (int i=length; i>0; i--) {                  if (pos[0].x == pos[i].x && pos[0].y == pos[i].y) {                      alive = false;                  }              }              if (pos[0].y >= BOARD\_HEIGHT)                  alive = false;              if (pos[0].y < 0)                  alive = false;              if (pos[0].x >= BOARD\_WIDTH)                  alive = false;              if (pos[0].x < 0)                  alive = false;                if (!alive)                  timer.stop();                //move              for (int i=length; i>0; i--) {                  pos[i].x = pos[i-1].x;                  pos[i].y = pos[i-1].y;              }              if (move == LEFT)                  pos[0].x -= PIXEL;              if (move == RIGHT)                  pos[0].x += PIXEL;              if (move == UP)                  pos[0].y -= PIXEL;              if (move == DOWN)                  pos[0].y += PIXEL;              moved=true;              repaint();          }      } |
| AFTER |
| public void actionPerformed(ActionEvent e) {          if (alive) {              checkFood();              checkCollision();              move();              repaint();          }      }      public void checkFood(){          if ((pos[0].x == food\_x) && (pos[0].y == food\_y)) {              length++;              randomFood();              int d = DELAY-length;              if(d < 40) d = 40;              timer.setDelay(d);          }      }      public void checkCollision(){          for (int i=length; i>0; i--) {              if (pos[0].x == pos[i].x && pos[0].y == pos[i].y) {                  alive = false;              }          }          if (pos[0].y >= setup\_panel.getBOARD\_HEIGHT())              alive = false;          if (pos[0].y < 0)              alive = false;          if (pos[0].x >= setup\_panel.getBOARD\_WIDTH())              alive = false;          if (pos[0].x < 0)              alive = false;            if (!alive)              timer.stop();      }      public void move(){          for (int i=length; i>0; i--) {              pos[i].x = pos[i-1].x;              pos[i].y = pos[i-1].y;          }          if (move == LEFT)              pos[0].x -= setup\_panel.getPIXEL();          if (move == RIGHT)              pos[0].x += setup\_panel.getPIXEL();          if (move == UP)              pos[0].y -= setup\_panel.getPIXEL();          if (move == DOWN)              pos[0].y += setup\_panel.getPIXEL();          moved=true;      } |

|  |
| --- |
| SMELL 4, Bloaters(Large Class) dan Change\_preventers (divergent\_change) -> extract class  Penjelasan: dalam code Snake.java before class Panel merupakan sebuah class yang sangat besar dan dapat dikatakan memiliki lebih dari 1 responsibilitas (large class dan divergent change), seperti setup game awal, random food, gameover, paint component, dll. Oleh karena itu sudah seharus class ini di extract ke class lain sehingga hanya memiliki 1 responsibilitas saja. Pada kasus saya memisahkan randomfood menjadi suatu class sendiri sehingga mengurangi responsibilitas dari class Panel. |
| Before  class Panel extends JPanel implements ActionListener { //Bloaters(Large Class) dan Change\_preventers (divergent\_change)  -> extract class    private setupPanel setup\_panel;      private final int BOARD\_WIDTH;      private final int BOARD\_HEIGHT;      private final int PIXEL;      private final int MAX\_LENGTH;      private final int MAX\_POS;      private final int DELAY = 100;      private final int HEAD = 1;      private final int BODY = 2;      private final int FOOD = 3;      private final Position pos[];      private int length;      private int food\_x;      private int food\_y;        private int LEFT = 1;      private int RIGHT = 2;      private int UP = 4;      private int DOWN = 8;      private int move = RIGHT;      private boolean alive = true;      private boolean moved = false;      private Timer timer;      private Image body;      private Image food;      private Image head;      private String msg = "Game Over";      public Panel(int width, int height, int pixel) {          this.BOARD\_WIDTH = width;          this.BOARD\_HEIGHT = height;          this.PIXEL = pixel;          this.MAX\_LENGTH = BOARD\_WIDTH\*BOARD\_HEIGHT;          this.MAX\_POS = (BOARD\_WIDTH/PIXEL)-2;          this.pos = new Position[MAX\_LENGTH];          for(int i=0;i<pos.length;i++)              pos[i] = new Position();            addKeyListener(new TAdapter());          setBackground(Color.black);          setFocusable(true);          setPreferredSize(new Dimension(BOARD\_WIDTH, BOARD\_HEIGHT));          head = setupImages(HEAD);          body = setupImages(BODY);          food = setupImages(FOOD);          initGame();      }      private Image setupImages(int type) {          BufferedImage image = new BufferedImage(PIXEL, PIXEL, BufferedImage.TYPE\_INT\_RGB);          Graphics2D g2 = image.createGraphics();          switch(type) { //oo\_abuser(switch statements) dan dispensables(duplicate code) -> bikin subclasses              case BODY:                      g2.setPaint(Color.blue);                      g2.fillOval(0, 0, PIXEL, PIXEL);                      g2.dispose();                  break;              case HEAD:                      g2.setPaint(Color.red);                      g2.fillOval(0, 0, PIXEL, PIXEL);                      g2.dispose();                  break;              case FOOD:                      g2.setPaint(Color.green);                      g2.fillOval(2, 2, PIXEL-2, PIXEL-2);                      g2.setPaint(new Color(102,51,0));                      int [] x = {1, PIXEL/2, PIXEL/2};                      int [] y = {1, PIXEL/2, 1};                      g2.fillPolygon(x,y,3);                      g2.dispose();                  break;          }          return image;      }      private void initGame() {          length = 3;          for (int i=0; i<length; i++) {              pos[i].x = PIXEL\*5 - i \* 10;              pos[i].y = PIXEL\*5;          }            randomFood();          timer = new Timer(DELAY, this);          timer.start();      }      @Override      public void paintComponent(Graphics g) {          super.paintComponent(g);          if (alive) {              g.drawImage(food, food\_x, food\_y, this);              g.drawImage(head, pos[0].x, pos[0].y, this);              for (int i = 1; i < length; i++)                  g.drawImage(body, pos[i].x, pos[i].y, this);              Toolkit.getDefaultToolkit().sync();          } else {              gameOver(g);          }      }      private void gameOver(Graphics g) {          Font small = new Font("Helvetica", Font.BOLD, 14);          FontMetrics metr = getFontMetrics(small);          g.setColor(Color.white);          g.setFont(small);          g.drawString(msg, (BOARD\_WIDTH - metr.stringWidth(msg)) / 2, BOARD\_HEIGHT / 2);      }      private void randomFood() {          int r = (int) (Math.random() \* MAX\_POS)+1;          food\_x = ((r \* PIXEL));          r = (int) (Math.random() \* MAX\_POS)+1;          food\_y = ((r \* PIXEL));      }      @Override //long method and comments -> extract method      public void actionPerformed(ActionEvent e) {          if (alive) {              // check food              if ((pos[0].x == food\_x) && (pos[0].y == food\_y)) {                  length++;                  randomFood();                  int d = DELAY-length;                  if(d < 40) d = 40;                  timer.setDelay(d);              }              // checkCollision              for (int i=length; i>0; i--) {                  if (pos[0].x == pos[i].x && pos[0].y == pos[i].y) {                      alive = false;                  }              }              if (pos[0].y >= BOARD\_HEIGHT)                  alive = false;              if (pos[0].y < 0)                  alive = false;              if (pos[0].x >= BOARD\_WIDTH)                  alive = false;              if (pos[0].x < 0)                  alive = false;                if (!alive)                  timer.stop();                //move              for (int i=length; i>0; i--) {                  pos[i].x = pos[i-1].x;                  pos[i].y = pos[i-1].y;              }              if (move == LEFT)                  pos[0].x -= PIXEL;              if (move == RIGHT)                  pos[0].x += PIXEL;              if (move == UP)                  pos[0].y -= PIXEL;              if (move == DOWN)                  pos[0].y += PIXEL;              moved=true;              repaint();          }      }      private class TAdapter extends KeyAdapter {          @Override          public void keyPressed(KeyEvent e) {              if(!moved) return;              int key = e.getKeyCode();              switch(key) {                  case KeyEvent.VK\_LEFT:  if(move!=RIGHT) move = LEFT; break;                  case KeyEvent.VK\_RIGHT: if(move!=LEFT)  move = RIGHT; break;                  case KeyEvent.VK\_UP:    if(move!=DOWN)  move = UP; break;                  case KeyEvent.VK\_DOWN:  if(move!=UP)    move = DOWN; break;              }              moved = false;          }      }  } |
| AFTER |
| public Panel(int width, int height, int pixel) {          setup\_panel = new setupPanel(width, height, pixel);          this.MAX\_LENGTH = setup\_panel.getMAX\_LENGTH();          this.MAX\_POS = setup\_panel.getMAX\_POS();          this.pos = setup\_panel.getPos();            addKeyListener(new TAdapter());          setBackground(Color.black);          setFocusable(true);          setPreferredSize(new Dimension(setup\_panel.getBOARD\_WIDTH(), setup\_panel.getBOARD\_HEIGHT()));          setupImages setupimages = new setupImages(setup\_panel.getPIXEL());          head = setupimages.setupImagesHead(HEAD);          body = setupimages.setupImagesBody(BODY);          food = setupimages.setupImagesFood(FOOD);          initGame();          RandomFood randomFood = new RandomFood(setup\_panel.getPIXEL());          food\_x = randomFood.getFood();          food\_y = randomFood.getFood();      }  Jadi setelah method randomFood diextract menjadi suatu class, nanti tinggal dibuatkan object saja pada constructor class Panel. |

|  |
| --- |
| SMELL 3, OO\_abuser(switch statements)  Penjelasan, dalam code ini terdapat smell switch statement, jika dilihat pada method setupImages pada snake.java before kita dapat menemukan penggunakan switch statements yang kurang efektik karena switch tersebut digunakan untuk menentukan behaviour image tergantung dari tipe yang sesuai (HEAD,BODY,FOOD), oleh karena itu untuk menghilang penggunaan switch statement saya melakukan extract class untuk method setupimages dan switch statements tersebut diubah menjadi method2 dalam class baru setupImages.java |
| BEFORE |
| Snake.java      public Panel(int width, int height, int pixel) {          this.BOARD\_WIDTH = width;          this.BOARD\_HEIGHT = height;          this.PIXEL = pixel;          this.MAX\_LENGTH = BOARD\_WIDTH\*BOARD\_HEIGHT;          this.MAX\_POS = (BOARD\_WIDTH/PIXEL)-2;          this.pos = new Position[MAX\_LENGTH];          for(int i=0;i<pos.length;i++)              pos[i] = new Position();            addKeyListener(new TAdapter());          setBackground(Color.black);          setFocusable(true);          setPreferredSize(new Dimension(BOARD\_WIDTH, BOARD\_HEIGHT));          head = setupImages(HEAD);          body = setupImages(BODY);          food = setupImages(FOOD);          initGame();      }      private Image setupImages(int type) {          BufferedImage image = new BufferedImage(PIXEL, PIXEL, BufferedImage.TYPE\_INT\_RGB);          Graphics2D g2 = image.createGraphics();          switch(type) {              case BODY:                      g2.setPaint(Color.blue);                      g2.fillOval(0, 0, PIXEL, PIXEL);                      g2.dispose();                  break;              case HEAD:                      g2.setPaint(Color.red);                      g2.fillOval(0, 0, PIXEL, PIXEL);                      g2.dispose();                  break;              case FOOD:                      g2.setPaint(Color.green);                      g2.fillOval(2, 2, PIXEL-2, PIXEL-2);                      g2.setPaint(new Color(102,51,0));                      int [] x = {1, PIXEL/2, PIXEL/2};                      int [] y = {1, PIXEL/2, 1};                      g2.fillPolygon(x,y,3);                      g2.dispose();                  break;          }          return image;      } |
| AFTER |
| Snake.java      public Panel(int width, int height, int pixel) {          setup\_panel = new setupPanel(width, height, pixel);          this.MAX\_LENGTH = setup\_panel.getMAX\_LENGTH();          this.MAX\_POS = setup\_panel.getMAX\_POS();          this.pos = setup\_panel.getPos();            addKeyListener(new TAdapter());          setBackground(Color.black);          setFocusable(true);          setPreferredSize(new Dimension(setup\_panel.getBOARD\_WIDTH(), setup\_panel.getBOARD\_HEIGHT()));          setupImages setupimages = new setupImages(setup\_panel.getPIXEL());          head = setupimages.setupImagesHead(HEAD);          body = setupimages.setupImagesBody(BODY);          food = setupimages.setupImagesFood(FOOD);          initGame();      }  --------------------------------------------------------------------------------------------------  setupImage.java  import java.awt.\*;  import java.awt.image.BufferedImage;  public class setupImages {      private int PIXEL;      BufferedImage image;      Graphics2D g2;      public setupImages(int PIXEL){          this.PIXEL = PIXEL;          this.image = new BufferedImage(this.PIXEL, this.PIXEL, BufferedImage.TYPE\_INT\_RGB);          this.g2 = image.createGraphics();      }      public Image setupImagesBody(int type) {          g2.setPaint(Color.blue);          g2.fillOval(0, 0, PIXEL, PIXEL);          g2.dispose();          return image;      }      public Image setupImagesHead(int type) {          g2.setPaint(Color.red);          g2.fillOval(0, 0, PIXEL, PIXEL);          g2.dispose();          return image;      }      public Image setupImagesFood(int type) {          g2.fillOval(2, 2, PIXEL-2, PIXEL-2);          g2.setPaint(new Color(102,51,0));          int [] x = {1, PIXEL/2, PIXEL/2};          int [] y = {1, PIXEL/2, 1};          g2.fillPolygon(x,y,3);          g2.dispose();          return image;      }  } |

|  |
| --- |
| SMELL 2, Bloaters(data\_clumps) -> membuat suatu object yang akan mewakiliki variable2 yang bersama2 terus.  Penjelasan : di dalam Snake.java ditemukan beberapa variable yang selalu berbarengan (BOARD\_WIDTH, BOARD\_HEIGHT, dan juga PIXEL) hal ini menyebabkan smell data clumps. Oleh karena itu lebih baik dibuatkan saja suatu object dalam hal ini saya membuat class java baru yaitu setupPanel.java. sehingga pada akhirnya snake.java terlihat lebih rapih karena variable tersebut berubah menjadi suatu object. |
| BEFORE |
| Snake.java      private final int BOARD\_WIDTH;      private final int BOARD\_HEIGHT;      private final int PIXEL;      private final int MAX\_LENGTH;      private final int MAX\_POS;      private final int DELAY = 100;      private final int HEAD = 1;      private final int BODY = 2;      private final int FOOD = 3;      private final Position pos[];  public Panel(int width, int height, int pixel) {          this.BOARD\_WIDTH = width;          this.BOARD\_HEIGHT = height;          this.PIXEL = pixel;          this.MAX\_LENGTH = BOARD\_WIDTH\*BOARD\_HEIGHT;          this.MAX\_POS = (BOARD\_WIDTH/PIXEL)-2;          this.pos = new Position[MAX\_LENGTH];          for(int i=0;i<pos.length;i++)              pos[i] = new Position();            addKeyListener(new TAdapter());          setBackground(Color.black);          setFocusable(true);          setPreferredSize(new Dimension(BOARD\_WIDTH, BOARD\_HEIGHT));          head = setupImages(HEAD);          body = setupImages(BODY);          food = setupImages(FOOD);          initGame();      } |
| AFTER |
| private setupPanel setup\_panel;      private final int MAX\_LENGTH;      private final int MAX\_POS;      private final int DELAY = 100;      private final int HEAD = 1;      private final int BODY = 2;      private final int FOOD = 3;      private final Position pos[];      public Panel(int width, int height, int pixel) {          setup\_panel = new setupPanel(width, height, pixel);          this.MAX\_LENGTH = setup\_panel.getMAX\_LENGTH();          this.MAX\_POS = setup\_panel.getMAX\_POS();          this.pos = setup\_panel.getPos();            addKeyListener(new TAdapter());          setBackground(Color.black);          setFocusable(true);          setPreferredSize(new Dimension(setup\_panel.getBOARD\_WIDTH(), setup\_panel.getBOARD\_HEIGHT()));          head = setupImages(HEAD);          body = setupImages(BODY);          food = setupImages(FOOD);          initGame();      }  ----------------------------------------------------------------------------------------------------------------  setupPanel.java  public class setupPanel {      private final int BOARD\_WIDTH;      private final int BOARD\_HEIGHT;      private final int PIXEL;      private Position pos[];      public setupPanel(int BOARD\_WIDTH, int BOARD\_HEIGHT, int PIXEL){          this.BOARD\_WIDTH = BOARD\_WIDTH;          this.BOARD\_HEIGHT = BOARD\_HEIGHT;          this.PIXEL = PIXEL;      }      public int getBOARD\_WIDTH() {          return BOARD\_WIDTH;      }      public int getBOARD\_HEIGHT() {          return BOARD\_HEIGHT;      }      public int getPIXEL() {          return PIXEL;      }      public int getMAX\_LENGTH() {          return this.BOARD\_WIDTH \* this.BOARD\_HEIGHT;      }      public int getMAX\_POS() {          return (this.BOARD\_WIDTH/this.PIXEL)-2;      }      public Position[] getPos() {          this.pos = new Position[getMAX\_LENGTH()];          for(int i=0;i<pos.length;i++)              pos[i] = new Position();          return pos;      }  } |

|  |
| --- |
| SMELL 1, dispensables (Dead Code) -> delete code  Penjelasan: import java.awt.geom tidak digunakan didalam code sehingga dapat dikatan dead code dan menyelesaikannya hanya tinggal didelete saja codenya. Selain import tersebut pada snake.java ditemukan lagi deadcode pada method setupImagesFood yaitu pada g2.setPaint. g2.setPaint(Color.green) tidak akan jalan karena pasti akan teroveride dengan g2.setPaint(new Color(102,51,0). |
| BEFORE |
| import java.awt.geom.\*;  public Image setupImagesFood(int type) {          g2.setPaint(Color.green);          g2.fillOval(2, 2, PIXEL-2, PIXEL-2);          g2.setPaint(new Color(102,51,0));          int [] x = {1, PIXEL/2, PIXEL/2};          int [] y = {1, PIXEL/2, 1};          g2.fillPolygon(x,y,3);          g2.dispose();          return image;      } |
| AFTER |
| 1. Delete code import tersebut karena tidak digunakan. 2. Memilihi setPaint mana yang ingin digunakan , pada kasus ini saya pilih g2.setPaint(new Color(102,51,0).   public Image setupImagesFood(int type) {          g2.fillOval(2, 2, PIXEL-2, PIXEL-2);          g2.setPaint(new Color(102,51,0));          int [] x = {1, PIXEL/2, PIXEL/2};          int [] y = {1, PIXEL/2, 1};          g2.fillPolygon(x,y,3);          g2.dispose();          return image;      } |