LAPORAN TUGAS BESAR 2

IF2210/Pemrograman Berorientasi Objek

ArkavQuarium

Dipersiapkan oleh:

K02D - AkadQuarium

13516014 - Renjira Naufhal Dhiaegana

13516035 - Muhammad Sulthan Adhipradhana

13516074 - Muhammad Abdullah Munir

13516104 - Muhammad Alfian Rasyidin

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

Daftar Isi

Screenshot Program 56

1 Ringkasan

Persoalan yang diselesaikan dalam tugas besar kali ini yaitu membuat program permainan mirip dengan Insaniquarium yang diberi nama dengan ArkavQuarium. Permainan ini ini dijalankan oleh satu orang pemain dengan tujuan menang dari permainan ini memperoleh tiga telur. Telur didapatkan dengan melakukan pembelian dengan koin yang dimiliki oleh pemain. Koin dihasilkan oleh ikan-ikan.

Terdapat 2 jenis ikan dalam permainan ini yang memiliki karakteristik seperti berikut:

1. Guppy

Guppy dapat dibeli dengan koin yang dimiliki oleh pemain. Terdapat 3 tahap pertumbuhan dari ikan ini. Makanan dari ikan ini yaitu makanan ikan yang dibeli dengan koin dan diberikan oleh pengguna secara manual. Ikan ini akan mengejar makanan ikan saat ia kelaparan, sedangkan akan bergerak secara acak jika tidak sedang dalam keadaan lapar. Ikan ini juga akan secara teratur menghasilkan koin yang memiliki nilai yang berbeda-beda tergantung dengan level pertumbuhan ikan tersebut saat menghasilkan koin.

2. Piranha

Secara umum, ikan ini mirip dengan ikan Guppy dalam spesifikasi pembelian, makan, gerak, dan lainnya. Namun, makanan ikan ini bukanlah makanan ikan, namun ia memakan ikan Guppy yang dekat dengannya. Ia juga hanya makan saat keadaan lapar saja. Ikan ini akan menghasilkan koin saat ia makan, dan koin dilepaskan seketika itu juga.

Jika tidak terdapat makanan ikan saat keadaan ikan sedang lapar, maka ikan akan mati. Untuk mengumpulkan koin-koin tersebut, terdapat satu objek akuarium lain yaitu siput yang akan bergerak ke arah koin berada. Siput ini sudah ada saat permainan dimulai dan tidak pernah mati selama permainan masih berlangsung.

Laporan ini dibuat dengan tujuan untuk menjelaskan program yang telah kami telah buat, baik struktur data, desain kelas, dan implementasi program. Struktur laporan ini yaitu penjelasan deskripsi umum tugas, penjelasan tambahan spesifikasi tugas baik yang merupakan syarat wajib atau syarat opsional dari tugas, rancangan kelas yang merupakan desain kelas yang telah kami

buat dan penjelasan perubahan kelas dari tugas sebelumnya, program utama, *test-script* untuk keperluan pengecekan kebenaran implementasi kelas, dan pembagian kerja dalam kelompok.

Simpulan dari tugas ini ialah tugas ini melatih mahasiswa untuk dapat merancang dan mengimplementasikan ilmu-ilmu konsep pemograman berorientasi objek serta penggunaan struktur data yang telah diajarkan pada mata kuliah sebelumnya.

2 Penjelasan Tambahan Spesifikasi Tugas

Berikut beberapa tambahan penjelasan spesifikasi tugas yang belum rinci yang didapatkan dari hasil asistensi pertama:

1. Tugas besar dua secara umum spesifikasinya sama dengan tugas besar 1.
2. LinkedList harus dibuat sendiri seperti pada tugas besar 1, tidak boleh menggunakan *library* yang disediakan oleh Java.

2.1 Visualisasi

- Terdapat latar belakang yang mirip dengan permainan aslinya.
- Guppy menghasilkan koin, sedangkan Piranha menghasilkan *diamond.*
- Ikan akan bertambah besar seiring level pertumbuhannya naik.
- Semua objek akuarium bergerak ke depan, sehingga dapat berputar arah.
- Terdapat latar musik yang mengiringi permainan.

2.2 Main Menu

- Pada saat program pertama kali dijalankan, terdapat fitur menu.
- Permainan dimulai saat pemain menekan tombol mulai.
- pemain dapat melakukan *pause* atau jeda saat permainan berlangsung.
- Terdapat menu bar di bagian atas, yang memungkinkan pemain memilih jenis ikan yang akan dibeli.

2.3 Save dan *Load*

- Menyimpan keadaan terakhir dalam permainan.

- Melakukan *load* dari save.txt dengan menekan tombol angka “1”.

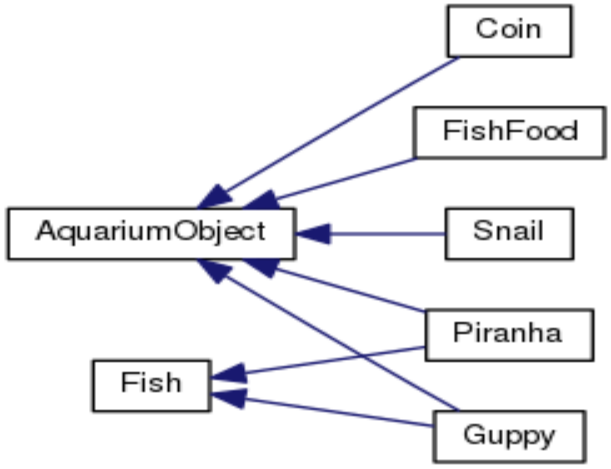
2.4 Penentuan Posisi Makanan

- Makanan selalu jatuh dari bagian atas akuarium.
- Posisi x dari makanan tersebut bergantung dimana posisi kursor sedang berada saat makanan dilepaskan.

2.5 Mengambil Koin dengan *Mouse*

- Pemain dapat mengambil koin atau *diamond* sebelum atau sesudah koin atau *diamond* tersebut sampai ke dasar akuarium.
- Pengambilan koin selain dilakukan oleh Snail, pemain juga dapat mengambilnya secara langsung dengan menggunakan mouse.

3 Rancangan Kelas



Gambar 1. Diagram Kelas

3.1 Perubahan dari Tugas Besar

No	Nama Kelas	Nama Method	Uraian
1	Node	private T data; private Node<T> next; public Node(final T pData); public void setNext(final Node<T>	Pada tugas kecil sebelumnya, kami berencana untuk mengimplementasikan LinkedList

		pNext);
		public void setData(final T pData);
		public T getData();
		public Node<T> getNext();
			secara rekursif, yaitu LinkedList to LinkedList.
			
			Perubahan yang terjadi yaitu LinkedList akan menyimpan first yaitu node pertamanya, dan data berikutnya akan disimpan di Node yaitu di atribut Data.

2	LinkedList	private int count;
		private Node<T> first;
		private T val;
		
		public T get(final int idx);
		public int getCount();
		private int find(final T pVal);
		public boolean isEmpty();
		void add(final T pVal);
		public void remove(final T node);	

3	Piranha	`private double targetX;`
		`private double targetY;`
		`private Guppy targetFood;`
		`private int hunger;`
		`private long lastHungerTime;`
		`private long lastMoveTime;`
		`private static LinkedList<Coin> listCoin;`
		`private static LinkedList<Guppy> listGuppy;`
		`private static LinkedList<Piranha> listPiranha;`
		`private static final int HUNGER_TIME = 30;`
		`private static final int MAX_HUNGER = 60;`
		`private static final double VELOCITY = 1;`
		`private static final int GUPPY_PRICE = 15;`
		`private static final double RADIUS = 40;`
		`private static BufferedImage piranhaLeft;`
		`private static BufferedImage piranhaRight;`
			Ada beberapa tambahan method public untuk keperluan kelas tersebut.
			
			Terdapat beberapa tambahan konstanta dari kelas ini juga yang belum dideklarasikan pada tugas kecil sebelumnya.

		void setHunger(final int pHunger);
		static void setPiranhaLeft(final BufferedImage pPiranhaLeft);
		static void setPiranhaRight(final BufferedImage pPiranhaRight);
		static void setListCoin(final LinkedList<Coin> pListCoin);
		static void setListGuppy(final LinkedList<Guppy> pListGuppy);
		static void setListPiranha(final LinkedList<Piranha> pListPiranha);
		double getRadius();
		int getHunger();
		static LinkedList<Coin> getListCoin();
		static LinkedList<Guppy> getListGuppy();
		static LinkedList<Piranha> getListPiranha();
		boolean isHungry();
		public void eat();
		public void dropCoin();
		public void move(final Graphics g);	

4	Snail	`private Node<Integer> money;`
		`private static BufferedImage snailLeft;`
		`private static BufferedImage snailRight;`
		`private static final double RADIUS = 30;`
		`private static final double VELOCITY = 1;`
		`private static LinkedList<Coin> listCoin;`
		
		`public static void setListCoin(final LinkedList<Coin> pListCoin);`
		`public void setMoney(final Node<Integer> pMoney);`
		`public static void setSnailLeft(final BufferedImage pSnailLeft);`
		`public static void setSnailRight(final BufferedImage pSnailRight);`
		`public double getRadius();`
		`public static LinkedList<Coin> getListCoin();`
		`public void grabCoin(final Coin coin);`
		`public void move(final Graphics g);`
			Terdapat beberapa method public pada kelas ini untuk keperluan penggunaan kelas. Selain itu, terdapat penambahan beberapa method seperti grabCoin.
			
			Ada tambahan atribut kelas ini yaitu konstanta yang baru didefinisikan.

5	Guppy	`private int state;`
		`private int hunger;`
		`private int timesEaten;`
		`private long lastMoveTime;`
		`private long lastDropTime;`
		`private long lastHungerTime;`
			Terdapat beberapa penambahan method untuk kelas ini dan mendefinisikan kontanta baru

```
private double targetX;
private double targetY;
private FishFood targetFood;
private static LinkedList<Coin>
listCoin;
private static LinkedList<Guppy>
listGuppy;
private static
LinkedList<FishFood>
listFishFood;
private static final int
DROP_TIME = 15;
private static final int
MAX_HUNGER = 60;
private static final int
HUNGER_TIME = 30;
private static final int
COIN_DROP_VALUE = 7;
private static final double
RADIUS = 40;
private static final double
VELOCITY = 1;
private static BufferedImage
stateOneGuppyLeft;
private static BufferedImage
stateOneGuppyRight;
private static BufferedImage
stateTwoGuppyLeft;
private static BufferedImage
stateTwoGuppyRight;
private static BufferedImage
stateThreeGuppyLeft;
private static BufferedImage
stateThreeGuppyRight;

public static void setListCoin(
final LinkedList<Coin>
pListCoin);
public static void setListGuppy(
final LinkedList<Guppy>
pListGuppy);
public static void
setListFishFood(
final LinkedList<FishFood>
```

		pListFishFood);
		public static void
		setStateOneGuppyLeft(
		final
		BufferedImagepStateOneGuppyLeft)
		;
		public static void
		setStateOneGuppyRight(
		final
		BufferedImagepStateOneGuppyRight
		);
		public static void
		setStateTwoGuppyLeft(final
		BufferedImagepStateTwoGuppyLeft)
		;
		public static void
		setStateTwoGuppyRight(final
		BufferedImagepStateTwoGuppyRight
		);
		public static void
		setStateThreeGuppyLeft(final
		BufferedImagepStateThreeGuppyLef
		t);
		public static void
		setStateThreeGuppyRight(final
		BufferedImagepStateThreeGuppyRig
		ht);
		void setHunger(final int
		pHunger);
		void setState(final int pState);
		void setTimesEaten(final int
		pTimesEaten);
		double getRadius();
		int getHunger();
		int getTimesEaten();
		int getState();
		LinkedList<Coin> getListCoin();
		LinkedList<FishFood>
		getListFishFood();
		LinkedList<Guppy>
		getListGuppy();
		boolean equals(final Guppy
		pGuppy);
		boolean isHungry();	

		public void eat();
		public void dropCoin();
		void move(final Graphics g);
		private void drawGuppy(final Graphics g, final BufferedImage stateOneGuppy,
		final BufferedImage stateTwoGuppy,
		final BufferedImage stateThreeGuppy);	

6	FishFood	private static final double VELOCITY = 0.5;
		private static final double RADIUS = 5;
		private static final double LOW_BOUNDARY = 600;
		private static BufferedImage image;
		private static LinkedList<FishFood> listFishFood;
		
		public static void setImage(final BufferedImage pImage);
		public static void setListFishFood(
		final LinkedList<FishFood> pListFishFood);
		public double getRadius();
		public static double getVelocity();
		public static LinkedList<FishFood> getListFishFood();
		public void move(final Graphics g);
			Terdapat beberapa tambahan *method* pada kelas ini dan mendefinisikan atribut-atribut konstanta yang belum didefinisikan di tugas sebelumnya

7	Coin	private final int value; private static BufferedImage imageCoin; private static BufferedImage imageDiamond; private static LinkedList<Coin>	Terdapat beberapa tambahan *method* pada kelas ini dan mendefinisikan atribut-atribut konstanta yang belum didefinisikan di tugas sebelumnya

		listCoin;
		private static final int MIN_DIAMOND = 20;
		private static final int LOW_BOUNDARY = 575;
		private static final double RADIUS = 25;
		private static final double VELOCITY = 0.5;
		
		public static void setImageCoin(final BufferedImage pImageCoin);
		public static void setListCoin(final LinkedList<Coin> pListCoin);
		double getRadius();
		int getValue();
		public static LinkedList<Coin> getListCoin();
		public void move(final Graphics g);	

8	AquariumObject	private double x, y;
		private long lastLoopTime;
		
		public void setLastLoopTime(final long pLastLoopTime);
		public void setXi(final double pX);
		public void setYi(final double pY);
		public double getXi();
		public double getYi();
		double getDistance(final AquariumObject aquariumObject);
		long getTimeNow();
		public long getLastLoopTime();
		boolean isIntersect(final AquariumObject aquariumObject);
		boolean isIntersect(final double pX, final double pY, final double r);
			Terdapat beberapa tambahan *method* pada kelas ini dan mendefinisikan atribut-atribut konstanta yang belum didefinisikan di tugas sebelumnya

		`void draw(final Graphics g, final BufferedImage img);`
		`abstract void move(Graphics g);;`
		`abstract double getRadius();`	

9	Aquarium	`private int xClick;`
		`private int yClick;`
		`private int eggState;`
		`private boolean run;`
		`private boolean paused;`
		`private boolean accessSave;`
		`private Node<Integer> money;`
		`private static Font fontType;`
		
		`private Snail snail;`
		`private LinkedList<Coin> listCoin;`
		`private LinkedList<Guppy> listGuppy;`
		`private LinkedList<Piranha> listPiranha;`
		`private LinkedList<FishFood> listFishFood;`
		
		`private static BufferedImage background;`
		`private static BufferedImage eggPictureOne;`
		`private static BufferedImage eggPictureTwo;`
		`private static BufferedImage winBackground;`
		`private static BufferedImage menuBackground;`
		`private static BufferedImage loseBackground;`
		`private static BufferedImage eggPictureThree;`
		
		`private static final Integer EGG_Y = 5;`
		`private static final Integer EGG_X = 270;`
		`private static final Integer`
			Terdapat beberapa tambahan *method* pada kelas ini dan mendefinisikan atribut-atribut konstanta yang belum didefinisikan di tugas sebelumnya

		MONEY_Y = 85; private static final Integer MONEY_X = 940; private static final Integer INIT_MONEY = 25; private static final Integer GUPPY_VALUE = 15; private static final Integer STR_STATE_X = 400; private static final Integer STR_STATE_Y = 400; private static final Integer PIRANHA_VALUE = 30; private static final Integer FISHFOOD_VALUE = 5; private static final Integer INIT_SNAIL_X = 320; private static final Integer INIT_SNAIL_Y = 575; private static final Integer FIRST_EGG_VALUE = 100; private static final Integer SECOND_EGG_VALUE = 150; private static final Integer THIRD_EGG_VALUE = 250; public Aquarium(); static void loadData(final Aquarium pAquarium); static void saveData(final Aquarium pAquarium); public void setxClick(final int pXClick); public void setyClick(final int pYClick); public void setEggState(final int pEggState); public static void setEggPictureOne(final BufferedImage pEggPictureOne); public static void setEggPictureTwo(final BufferedImage pEggPictureTwo); public static void	

		setEggPictureThree(final BufferedImage pEggPictureThree);
		public static void setBackground(final BufferedImage pBackground);
		public static void setWinBackground(final BufferedImage pWinBackground);
		public static void setLoseBackground(final BufferedImage pLoseBackground);
		public static void setMenuBackground(final BufferedImage pMenuBackground);
		public static void setFontType(final Font pFontType);
		public LinkedList<Guppy> getListGuppy();
		public LinkedList<Piranha> getListPiranha();
		public LinkedList<Coin> getListCoin();
		public LinkedList<FishFood> getListFishFood();
		public Snail getSnail();
		public Node<Integer> getMoney();
		public int getEggState();
		public void mouseClicked(final MouseEvent e);
		public void mousePressed(final MouseEvent e);
		public void mouseReleased(final MouseEvent e);
		public void mouseEntered(final MouseEvent e);
		public void mouseExited(final MouseEvent e);
		private void moveAll(final Graphics g);
		private void drawWin(final Graphics g);
		private void drawLose(final	

		Graphics g); public void paint(final Graphics g);	

10	Main	private static JFrame jFrame;
		private static AudioInputStream audioIn;
		private static final int WINDOW_WIDTH = 1100;
		private static final int WINDOW_HEIGHT = 720;
		
		private Main();
		static JFrame initGameFrame();
		static void frameUpdate();
		static void loadResources();
		public static void main(final String[] args)
			Kelas ini merupakan kelas yang baru dari tugas sebelumnya. Kelas ini berguna untuk mengontrol kelas-kelas lain selama program berjalan, atau sering disebut sebagai *controller.*

4 Rincian Kelas

4.1 Aquarium

```
/**
 * Aquarium class.
 * This class instantiates Aquarium.
 */
private int xClick;
private int yClick;
private int eggState;
private boolean run;
private boolean paused;
private boolean accessSave;
private Node<Integer> money;
private static Font fontType;

private Snail snail;
private LinkedList<Coin> listCoin;
private LinkedList<Guppy> listGuppy;
private LinkedList<Piranha> listPiranha;
```

```
private LinkedList<FishFood> listFishFood;

private static BufferedImage background;
private static BufferedImage eggPictureOne;
private static BufferedImage eggPictureTwo;
private static BufferedImage winBackground;
private static BufferedImage menuBackground;
private static BufferedImage loseBackground;
private static BufferedImage eggPictureThree;

private static final Integer EGG_Y = 5;
private static final Integer EGG_X = 270;
private static final Integer MONEY_Y = 85;
private static final Integer MONEY_X = 940;
private static final Integer INIT_MONEY = 25;
private static final Integer GUPPY_VALUE = 15;
private static final Integer STR_STATE_X = 400;
private static final Integer STR_STATE_Y = 400;
private static final Integer PIRANHA_VALUE = 30;
private static final Integer FISHFOOD_VALUE = 5;
private static final Integer INIT_SNAIL_X = 320;
private static final Integer INIT_SNAIL_Y = 575;
private static final Integer FIRST_EGG_VALUE = 100;
private static final Integer SECOND_EGG_VALUE = 150;
private static final Integer THIRD_EGG_VALUE = 250;

/**
 * Instantiates a new Aquarium.
 */
public Aquarium() {}

/**
 * This method is used to load data that has been saved to resume the game.
 *
 * @param pAquarium is the Aquarium
 */
static void loadData(final Aquarium pAquarium) {}

/**
 * This method is used to save the current game.
 *
```

```
 * @param pAquarium is Aquarium
 */
static void saveData(final Aquarium pAquarium) {}

/**
 * Set x clicked coordinate position.
 *
 * @param pXClick is the abscissa
 */
public void setxClick(final int pXClick) {}

/**
 * Set y clicked coordinate position.
 *
 * @param pYClick is the ordinate
 */
public void setyClick(final int pYClick) {}

/**
 * Set egg state value.
 *
 * @param pEggState egg state
 */
public void setEggState(final int pEggState) {}

/**
 * Set first egg picture.
 *
 * @param pEggPictureOne is the BufferedImage
 */
public static void setEggPictureOne(final BufferedImage pEggPictureOne) {}

/**
 * Set second egg picture.
 *
 * @param pEggPictureTwo is the BufferedImage
 */
public static void setEggPictureTwo(final BufferedImage pEggPictureTwo) {}

/**
 * Set third egg picture.
```

```
 *
 * @param pEggPictureThree is the BufferedImage
 */
public static void setEggPictureThree(final BufferedImage pEggPictureThree)
{}

/**
 * Set window background when the game is running.
 *
 * @param pBackground is the BufferedImage
 */
public static void setBackground(final BufferedImage pBackground) {}

/**
 * Set window background when the player wins.
 *
 * @param pWinBackground is the BufferedImage
 */
public static void setWinBackground(final BufferedImage pWinBackground) {}

/**
 * Set the window background when the player loses.
 *
 * @param pLoseBackground is the BufferedImage
 */
public static void setLoseBackground(final BufferedImage pLoseBackground) {}

/**
 * Set the window menu background.
 *
 * @param pMenuBackground is the BufferedImage
 */
public static void setMenuBackground(final BufferedImage pMenuBackground) {}

/**
 * Set the font type.
 *
 * @param pFontType is the Font
 */
public static void setFontType(final Font pFontType) {}
```

```
/**
 * Get list of Guppy.
 *
 * @return LinkedList<Guppy>
 */
public LinkedList<Guppy> getListGuppy() {}

/**
 * Get list of Piranha.
 *
 * @return LinkedList<Piranha>
 */
public LinkedList<Piranha> getListPiranha() {}

/**
 * Get list of Coin.
 *
 * @return LinkedList<Coin>
 */
public LinkedList<Coin> getListCoin() {}

/**
 * Get list of FishFood.
 *
 * @return LinkedList<FishFood>
 */
public LinkedList<FishFood> getListFishFood() {}

/**
 * Get snail.
 *
 * @return snail
 */
public Snail getSnail() {}

/**
 * Get money.
 *
 * @return money
 */
public Node<Integer> getMoney() {}
```

```
/**
 * Get egg state.
 *
 * @return eggState
 */
public int getEggState() {}

/**
 * Actions when the mouse is clicked.
 *
 * @param e is the MouseEvent
 */
@Override
public void mouseClicked(final MouseEvent e) { }

/**
 * Actions when the mouse is pressed.
 *
 * @param e is the MouseEvent
 */
@Override
public void mousePressed(final MouseEvent e) {}

/**
 * Actions when the mouse is released.
 *
 * @param e is the MouseEvent
 */
@Override
public void mouseReleased(final MouseEvent e) { }

/**
 * Actions when the mouse is entered.
 *
 * @param e is the MouseEvent
 */
@Override
public void mouseEntered(final MouseEvent e) { }

/**
```

```
 * Actions when the mouse id exited.
 *
 * @param e is the MouseEvent
 */
@Override
public void mouseExited(final MouseEvent e) { }

/**
 * Invoke all object in Aquarium to move.
 *
 * @param g is Graphics
 */
private void moveAll(final Graphics g) {}

/**
 * Drawing win condition.
 *
 * @param g Graphics
 */
private void drawWin(final Graphics g) {}

/**
 * Drawing lose condition.
 *
 * @param g Graphics
 */
private void drawLose(final Graphics g) {}

/**
 * Draw aquarium objects while the game is running.
 *
 * @param g is Graphics
 */
@Override
public void paint(final Graphics g) {}
```

4.2 AquariumObject

```
 /**
 * Class AquariumObject.
 * This class is used as the abstraction class for another
```

```
 * aquarium objects for example: Guppy, Piranha, Snail, FishFood, etc
 */
private double x, y;
private long lastLoopTime;

/**
 * Set last loop time
 *
 * @param pLastLoopTime is the long type number that indicates time
 */
public void setLastLoopTime(final long pLastLoopTime) {}

/**
 * Set the abscissa
 *
 * @param pX is the abscissa
 */
public void setXi(final double pX) {}

/**
 * Set the ordinate
 *
 * @param pY is the ordinate
 */
public void setYi(final double pY) {}

/**
 * Get abscissa
 *
 * @return abscissa
 */
public double getXi() {}

/**
 * Get ordinate
 *
 * @return ordinate
 */
public double getYi() {}

/**
```

```
 * This method calculates the distance between two objects
 * Distance is defined as the square of the sum of the difference
 * between x (abscissa values) and y (ordinate values) of both objects
 *
 * @param aquariumObject is the AquariumObject
 * @return distance
 */
double getDistance(final AquariumObject aquariumObject) {}

/**
 * Get the current time
 *
 * @return current time in millisecond.
 */
long getTimeNow() {}

/**
 * Get last loop time
 *
 * @return last loop time
 */
public long getLastLoopTime() {}

/**
 * This method will check whether these both objects intersect
 * Intersect is defined as if square of the difference between x and y
 * position of both object is less than or equal to square of its radius
 *
 * @param aquariumObject is the AquariumObject
 * @return TRUE if both object intersect, otherwise FALSE
 */
boolean isIntersect(final AquariumObject aquariumObject) {}

/**
 * This method will check whether these two point intersect
 * Intersect is defined as if the square of the difference between
 * x and y coordinate is less than or equal to square of its radius
 *
 * @param pX is the abscissa
 * @param pY is the ordinate
 * @param r is the radius
```

```
 * @return TRUE if both points intersect, otherwise FALSE
 */
boolean isIntersect(final double pX, final double pY, final double r) {}

/**
 * This method will draw the aquarium object
 * on the aquarium while the game is running
 *
 * @param g is Graphics
 * @param img is BufferedImage
 */
void draw(final Graphics g, final BufferedImage img) {}

/**
 * Abstract method to move the object
 *
 * @param g is the Graphics
 */
abstract void move(Graphics g);

/**
 * Abstract method to get the radius
 *
 * @return radius
 */
abstract double getRadius();
```

4.3 Coin

```
 /**
 * Class Coin.
 * This class is derived from AquariumObject and used for
 * instantiate coins in the game in order to increase money.
 * Furthermore, coins are produced by all derived class of Fish
 */
private final int value;
private static BufferedImage imageCoin;
private static BufferedImage imageDiamond;
private static LinkedList<Coin> listCoin;
private static final int MIN_DIAMOND = 20;
private static final int LOW_BOUNDARY = 575;
```

```
private static final double RADIUS = 25;
private static final double VELOCITY = 0.5;

/**
 * Instantiates a new Coin.
 *
 * @param x is the abscissa
 * @param y is the ordinate
 * @param pValue is the coin value
 */
Coin(final double x, final double y, final int pValue) {}

/**
 * Set image of coin.
 *
 * @param pImageCoin is the BufferedImage
 */
public static void setImageCoin(final BufferedImage pImageCoin) {}

/**
 * Set image of diamond.
 *
 * @param pImageDiamond is the BufferedImage
 */
public static void setImageDiamond(final BufferedImage pImageDiamond) {}

/**
 * Set list of coin.
 *
 * @param pListCoin is the list coin
 */
public static void setListCoin(final LinkedList<Coin> pListCoin) {}

@Override
double getRadius() {}

/**
 * Get coin's value
 *
 * @return coin's value
 */
```

```
int getValue() {}

/**
 * Get list of coin.
 *
 * @return the list of coin
 */
public static LinkedList<Coin> getListCoin() {}

@Override
public void move(final Graphics g) {}
```

4.4 Fish

```
 /**
 * Interface Fish.
 * This interface is used to defined the Fish object.
 */
/**
 * This method represents the action of eating
 */
void eat();

/**
 * This method used to drop the coin
 */
void dropCoin();
```

4.5 FishFood

```
 /**
 * Class FishFood.
 * This class is derived from AquariumObject.
 * It is used as the food of Guppy - a derived Fish object.
 * FishFood is bought by the player using the coin while the game is running.
 */
public class FishFood extends AquariumObject {
    private static final double VELOCITY = 0.5;
    private static final double RADIUS = 5;
    private static final double LOW_BOUNDARY = 600;
    private static BufferedImage image;
    private static LinkedList<FishFood> listFishFood;
```

```
    /**
     * Instantiate FishFood
     *
     * @param x is the abscissa
     * @param y is the ordinate
     */
    public FishFood(final double x, final double y) {}

    /**
     * Set image of FishFood.
     *
     * @param pImage is BufferedImage
     */
    public static void setImage(final BufferedImage pImage) {}

    /**
     * Set list of fish food.
     *
     * @param pListFishFood is the list of FishFood
     */
    public static void setListFishFood(final LinkedList<FishFood>
pListFishFood) {}

    @Override
    public double getRadius() {}

    /**
     * Get velocity
     *
     * @return velocity
     */
    public static double getVelocity() {}

    /**
     * Get list of FishFood.
     *
     * @return listFishFood list fish food
     */
    public static LinkedList<FishFood> getListFishFood() {}
```

```
    /**
     * FishFood is moving down as free fall.
     * After it reaches ground, it will disappear.
     *
     * @param g is the Graphics
     */
    @Override
    public void move(final Graphics g) {}
}
```

4.6 Guppy

```
 /**
* Class Guppy.
* This class is derived from AquariumObject.
* Guppy is an aquarium object than can be bought
* during the game using the coins that the player has.
* It also can drop coins in a regular interval of time with variety of value.
*/
private int state;
private int hunger;
private int timesEaten;
private long lastMoveTime;
private long lastDropTime;
private long lastHungerTime;
private double targetX;
private double targetY;
private FishFood targetFood;
private static LinkedList<Coin> listCoin;
private static LinkedList<Guppy> listGuppy;
private static LinkedList<FishFood> listFishFood;
private static final int DROP_TIME = 15;
private static final int MAX_HUNGER = 60;
private static final int HUNGER_TIME = 30;
private static final int COIN_DROP_VALUE = 7;
private static final double RADIUS = 40;
private static final double VELOCITY = 1;
private static BufferedImage stateOneGuppyLeft;
private static BufferedImage stateOneGuppyRight;
private static BufferedImage stateTwoGuppyLeft;
private static BufferedImage stateTwoGuppyRight;
```

```
private static BufferedImage stateThreeGuppyLeft;
private static BufferedImage stateThreeGuppyRight;

/**
 * Instantiate Guppy
 */
public Guppy() {}

/**
 * Set list of coins
 *
 * @param pListCoin is list of coins
 */
public static void setListCoin(final LinkedList<Coin> pListCoin) {}

/**
 * Set list of Guppy
 *
 * @param pListGuppy is list of Guppy
 */
public static void setListGuppy(final LinkedList<Guppy> pListGuppy) {}

/**
 * Set list of FishFood
 *
 * @param pListFishFood is list of FishFood
 */
public static void setListFishFood(final LinkedList<FishFood> pListFishFood)
{}

/**
 * Set image of Guppy.
 * Guppy is on first level and facing left
 *
 * @param pStateOneGuppyLeft is BufferedImage
 */
public static void setStateOneGuppyLeft(final BufferedImage
pStateOneGuppyLeft) {}

/**
 * Set image of Guppy.
```

```
 * Guppy is on first level and facing right
 *
 * @param pStateOneGuppyRight is BufferedImage
 */
public static void setStateOneGuppyRight(final BufferedImage
pStateOneGuppyRight) {}

/**
 * Set image of Guppy.
 * Guppy is on second level and facing left
 *
 * @param pStateTwoGuppyLeft is BufferedImage
 */
public static void setStateTwoGuppyLeft(final BufferedImage
pStateTwoGuppyLeft) {}

/**
 * Set image of Guppy.
 * Guppy is on second level and facing right
 *
 * @param pStateTwoGuppyRight is BufferedImage
 */
public static void setStateTwoGuppyRight(final BufferedImage
pStateTwoGuppyRight) {}

/**
 * Set image of Guppy.
 * Guppy is on third level and facing left
 *
 * @param pStateThreeGuppyLeft is BufferedImage
 */
public static void setStateThreeGuppyLeft(final BufferedImage
pStateThreeGuppyLeft) {}

/**
 * Set image of Guppy.
 * Guppy is on third level and facing right
 *
 * @param pStateThreeGuppyRight is BufferedImage
 */
```

```
public static void setStateThreeGuppyRight(final BufferedImage
pStateThreeGuppyRight) {}

/**
 * Set hunger
 *
 * @param pHunger is integer represents hunger
 */
void setHunger(final int pHunger) {}

/**
 * Set state
 *
 * @param pState is integer represents state
 */
void setState(final int pState) {}

/**
 * Set total times eaten
 *
 * @param pTimesEaten is integer represents times eaten
 */
void setTimesEaten(final int pTimesEaten) {}

/**
 * Get radius
 *
 * @return radius
 */
double getRadius() {}

/**
 * Get hunger
 *
 * @return hunger which is the time between now and the last eat time
 */
int getHunger() {}

/**
 * Get times eaten
 *
```

```
 * @return times eaten which is the number of times that the fish has eaten
 */
int getTimesEaten() {}

/**
 * Get state
 *
 * @return state
 */
int getState() {}

/**
 * Get list of coins
 *
 * @return list of coins
 */
LinkedList<Coin> getListCoin() {}

/**
 * Get list of fish food
 *
 * @return list of fish food
 */
LinkedList<FishFood> getListFishFood() {}

/**
 * Get list of guppy
 *
 * @return list of guppy
 */
LinkedList<Guppy> getListGuppy() {}

/**
 * This method checks whether both Guppy are the same object.
 * It is defined as if both x and y coordinates are the same.
 * @param pGuppy is Guppy
 * @return TRUE if both Guppy are the same object, otherwise FALSE
 */
boolean equals(final Guppy pGuppy) {}

/**
```

```
 * This method checks whether Guppy is hungry or not.
 * Hungry is defined as if hunger is less than or equal to HUNGER_TIME
 * @return TRUE if Guppy is hungry, otherwise FALSE
 */
boolean isHungry() {}

/**
 * This method will eat FishFood.
 * Guppy only eats when Guppy is hungry and
 * will try to reach the nearest food available.
 */
@Override
public void eat() {}

/**
 * This method will drop coin.
 * Coin will drop regularly based on certain interval.
 * Also, coin value calculated from the multiplication of COIN_DROP_VALUE and
state
 */
@Override
public void dropCoin() {}

/**
 * This method is used to move the object.
 * Guppy's movements are described as below:
 * 1. If Guppy is not hungry, Guppy will move randomly.
 * 2. If Guppy is hungry and there is no food available, then it moves
randomly.
 * 3. If Guppy is hungry and there is food available, then it moves towards
the nearest food.
 *
 * @param g is Graphics
 */
@Override
void move(final Graphics g) {}

/**
 * This method will draw Guppy
 * @param g is Graphics
 * @param stateOneGuppy is state of first level Guppy
```

```
 * @param stateTwoGuppy is state of second level Guppy
 * @param stateThreeGuppy is state of third level Guppy
 */
private void drawGuppy(final Graphics g, final BufferedImage stateOneGuppy,
final BufferedImage stateTwoGuppy,final BufferedImage stateThreeGuppy) {}
```

4.7 LinkedList

```
/**
 * Template class LinkedList.
 * This is a template class for LinkedList.
 *
 * @param <T> is the parameter type
 */
private int count;
private Node<T> first;
private T val;

/**
 * Instantiate LinkedList
 */
LinkedList() {
    this.count = 0;
}

/**
 * Get the object
 *
 * @param idx is the index
 * @return is the T
 */
public T get(final int idx) {}

/**
 * Get the total object in the LinkedList
 *
 * @return integer that represents the total number of object in the
LinkedList
 */
public int getCount() {}
```

```
/**
 * This methods is used for finding val in the list.\
 *
 * @param pVal is target
 * @return index of element in list if found.
 */
private int find(final T pVal) {}

/**
 * This method is used to check whether the LinkedList is empty or not
 *
 * @return TRUE if the list is empty, otherwise FALSE
 */
public boolean isEmpty() {}

/**
 * This method will add a new object to the LinkedList of this object
 *
 * @param pVal is the object.
 */
void add(final T pVal) {}

/**
 * This method will remove the object that is passed in its LinkedList
 *
 * @param node is the T.
 */
public void remove(final T node) {}
```

4.8 Node

```
/**
 * Class Node.
 * This is a template class for Node.
 *
 * @param <T> is the parameter type
 */

private T data;
private Node<T> next;
/**
```

```
  * Instantiate Node
 *
  * @param pData is the T
  */
public Node(final T pData) {}

/**
 * Set next
 *
 * @param pNext is the Node of T
 */
public void setNext(final Node<T> pNext) {}

/**
 * Set data
 *
 * @param pData is the T
 */
public void setData(final T pData) {}

 /**
  * Get data
  *
  * @return T
  */
 public T getData() {}

 /**
  * Get next
  *
  * @return the next Node
  */
public Node<T> getNext() {}
```

4.9 Piranha

```
 /**
 * Class Piranha.
 * This class implements Fish.
 * Piranha is an aquarium object than can be bought
 * during the game using the coins that the player has.
```

```
 * It also can drop coin directly after Piranha eats Guppy.
 */
private double targetX;
private double targetY;
private Guppy targetFood;
private int hunger;
private long lastHungerTime;
private long lastMoveTime;
private static LinkedList<Coin> listCoin;
private static LinkedList<Guppy> listGuppy;
private static LinkedList<Piranha> listPiranha;
private static final int HUNGER_TIME = 30;
private static final int MAX_HUNGER = 60;
private static final double VELOCITY = 1;
private static final int GUPPY_PRICE = 15;
private static final double RADIUS = 40;
private static BufferedImage piranhaLeft;
private static BufferedImage piranhaRight;

/**
 * Instantiate Piranha
 */
Piranha() {}

/**
 * Set hunger
 *
 * @param pHunger is integer represents hunger
 */
void setHunger(final int pHunger) {}

/**
 * Set image of Piranha facing left.
 *
 * @param pPiranhaLeft is BufferedImage
 */
static void setPiranhaLeft(final BufferedImage pPiranhaLeft) {}

/**
 * Set image of Piranha facing right.
 *
```

```
 * @param pPiranhaRight is BufferedImage
 */
static void setPiranhaRight(final BufferedImage pPiranhaRight) {}

/**
 * Set list of coins.
 *
 * @param pListCoin is LinkedList of Coin
 */
static void setListCoin(final LinkedList<Coin> pListCoin) {}

/**
 * Set list of Guppy
 *
 * @param pListGuppy is LinkedList of Guppy
 */
static void setListGuppy(final LinkedList<Guppy> pListGuppy) {}

/**
 * Set list of Piranha
 *
 * @param pListPiranha is LinkedList of Piranha
 */
static void setListPiranha(final LinkedList<Piranha> pListPiranha) {}

@Override
double getRadius() {}

/**
 * Get hunger
 *
 * @return hunger
 */
int getHunger() {}

/**
 * Get list of coins
 *
 * @return list of coins
 */
static LinkedList<Coin> getListCoin() {}
```

```
/**
 * Get list of Guppy
 *
 * @return list of Guppy
 */
static LinkedList<Guppy> getListGuppy() {}

/**
 * Get list of Piranha
 *
 * @return list of Piranha
 */
static LinkedList<Piranha> getListPiranha() {}

/**
 * This method checks whether Piranha is hungry or not
 * Hungry is defined as if hunger is less than or equal to HUNGER_TIME
 *
 * @return TRUE if Piranha is hungry, otherwise FALSE
 */
boolean isHungry() {}

/**
 * This method will eat Guppy.
 * Piranha only eats when Piranha is hungry and
 * will try to reach the nearest food available.
 */
@Override
public void eat() {}

/**
 * This method will drop coin.
 * Piranha will drop coin directly after eating
 * Also, coin value vary and be calculated as follows:
 * GUPPY_PRICE multiply by STATE_OF_GUPPY added by one
 */
@Override
public void dropCoin() {}

/**
```

```
 * This method is used to move the object.
 * Piranha's movements are described as below:
 * 1. If Piranha is not hungry, Piranha will move randomly.
 * 2. If Piranha is hungry and there is no food available, then it moves
randomly.
 * 3. If Piranha is hungry and there is food available, then it moves towards
the nearest food.
 *
 * @param g is Graphics
 */
@Override
public void move(final Graphics g) {}
```

4.10 Snail

```
 /**
 * Class Snail.
 * This class is derived from AquariumObject.
 * Snail is an aquarium object that collects coin.
 */

private Node<Integer> money;
private static BufferedImage snailLeft;
private static BufferedImage snailRight;
private static final double RADIUS = 30;
private static final double VELOCITY = 1;
private static LinkedList<Coin> listCoin;

/**
 * Instantiate Snail.
 *
 * @param pX is abscissa.
 * @param pY is ordinate.
 */
Snail(final double pX, final double pY) {}

/**
 * Set list of coin.
 *
 * @param pListCoin is list of coin
 */
```

```
public static void setListCoin(final LinkedList<Coin> pListCoin) {}

/**
 * Set money.
 *
 * @param pMoney is money
 */
public void setMoney(final Node<Integer> pMoney) {}

/**
 * Set image of Snail facing left.
 *
 * @param pSnailLeft is BufferedImage
 */
public static void setSnailLeft(final BufferedImage pSnailLeft) {}

/**
 * Set image of Snail facing right.
 *
 * @param pSnailRight is BufferedImage
 */
public static void setSnailRight(final BufferedImage pSnailRight) {}

@Override
public double getRadius() {}

/**
 * Get list of coin.
 *
 * @return list of coin
 */
public static LinkedList<Coin> getListCoin() {}

/**
 * This method is used to grab the coin
 *
 * @param coin is the coin
 */
public void grabCoin(final Coin coin)  {}

/**
```

```
 * This method is used to move the object.
 * Snail's movements are described as below:
 * 1. If there is coin available, Snail will move towards the nearest coin.
 * 2. If there is no coin available, Snail will just stay at its current
location.
 *
 * @param g is Graphics
 */
@Override
public void move(final Graphics g) {}
```

4.11 Main

```
/**
 * Class Main.
 * This is the controller class to run another classes during the game
 */
private static JFrame jFrame;
private static AudioInputStream audioIn;
private static final int WINDOW_WIDTH = 1100;
private static final int WINDOW_HEIGHT = 720;

/**
 * Default constructor.
 */
private Main() {}

/**
 * This method instantiates JFrame
 *
 * @return JFrame
 */
static JFrame initGameFrame() {}

/**
 * This method used to update the frame while the game is running
 */
static void frameUpdate() {}

/**
 * This method used to load resources, especially for the visual audio
purposes.
```

```
 */
static void loadResources() {}

/**
 * Main program.
 *
 * @param args is array string when execute program from command line
 */
public static void main(final String[] args) {}
```

5 Program Utama

Berikut merupakan algoritma dari program utama program ini:

1. initGameFrame(), mengembalikan *object Frame* dengan ukuran layar 1100x720.
2. loadResources(), mencoba untuk load resource yang dibutuhkan dalam game, seperti music, gambar, dan font.
3. Membuat timer untuk melakukan update frame dengan delay tertentu.
4. Memutar music yang telah di load.
5. Inisialisasi Aquarium baru, kemudian dimasukkan ke dalam *frame* yang telah dibuat sebelumnya.
6. Timer dinyalakan. Ketika *frame* di update aquarium akan menjalankan *method* paint di *class* tersebut.

6 Test Script

No.	Kelas	Nama File Driver	Fitur/Method yang diuji	Kasus Pengujian	Hasil yang Diharapkan	Hasil yang Keluar
1	Aquarium Object	AquariumObjectTest	setXi	setXi(3.0)	3.0	3.0
			setYi	setYi(3.0)	3.0	3.0
			getDistance	guppy1(x=3.0, y=3.0) guppy2(x=3.0, y=3.0) guppy1.getdistance(guppy)	0	0
			isIntersect	guppy1(x=3.0, y=3.0)	true	true

				guppy2(x=3.0, y=3.0) guppy1.isIntersect(guppy2)		
			isIntersect	guppy(x=3.0, y=3.0) guppy.isIntersect(3.0, 3.0, 40)	true	true
2	Coin	CoinTest	getRadius	RADIUS = 25 coin.getRadius()	25	25
			getValue	Coin(x=3.0, y -4.0, value = 50) coin.getValue()	50	50
			setListCoin	listCoin.get(0).getXi() listCoin.get(0).getYi() listCoin.get(o).getValue();	3.0 4.0 50	3.0 4.0 50
			getListCoin	Coin.getListCoin()	listCoin	listCoin
3	FishFood	FishFoodTest	setListFishFood	listFishFood.add() fishFood.x = 3.0 fishiFood.y = 4.0 listFishFood.getXi() listFishFood.getYi()	3.0 4.0	3.0 4.0
			getRadius	RADIUS = 5	5	5
			getVelocity()	VELOCITY = 0.5	0.5	0.5
			getListFishFood	FishFood.getListFishFood	listFishFood	listFishFood
4	Guppy	GuppyTest	setListGuppy	Guppy.setListGuppy(listGuppy)	listGuppy	listGuppy
			setHunger	guppy.setHunger(1000)	1000	1000
			setState	guppy.setTimesEaten(5)	5	5
			setLastHungerTime	guppy.setLastHungerTime(1000)	1000	1000
			setLastMoveTime	guppy.setLastMoveTime(1000)	1000	1000
			setLastDropTime	guppy.setLastDropTime	1000	1000
			getRadius()	RADIUS = 40 guppy.getRadius()	40	40

			equals	guppy.equals(guppy)	true	true
			isHungry	guppy.setHunger(29) guppy.setHunger(30) guppy.setHunger(31)	true true false	true true false
5	LinkedList	LinkedListTest	get()	linkedlist<Integer> linkedlist[0] = 2 linkedlist[1] = 1 .get(0).intValue()	2	2
			getCount()	linkedlist<Integer> linkedlist[0] = 2 linkedlist[1] = 1 .get(0).intValue()	2	2

			isEmpty()	listofInteger.isEmpty();
				listOfInteger.add(1);
				listOfInteger.add(2);
				listOfInteger.isEmpty();
					true
					false
						true
						false

			add()	listOfInteger.add(1);
				listOfInteger.add(2);
				listOfInteger.getCount();
				listOfInteger.remove(1);
				listOfInteger.getCount();
					2
					1
						2
						1

			remove()	listOfInteger.add(1);
				listOfInteger.add(2);
				listOfInteger.getCount();
				listOfInteger.remove(1);
				listOfInteger.getCount();
					2
					1
						2
						1

6	Node	NodeTest	setNext()	node1[Data] = 1 node2[Data] = 2 node1.setNext(node2);	node2	node2

				node1.getNext()		
			setData	node1.setData(2) node1.getData();	2	2
7	Piranha	PiranhaTest	setHunger	piranha.setHunger(3) piranha.getHunger()	3	3
			setListPiranha	listOfPiranha piranha.setListPiranha(listOfPiranha) piranha.getListPiranha()	listOfPiranha	listOfPiranha
			setLastMoveTime	piranha.setLastMoveTime(1000) piranha.getLastMoveTime()	1000	1000
			setLastHungerTime	piranha.setlastHungerTime(1000) piranha.getLastHungerTime()	1000	1000
			getRadius	piranha.getRadius()	40	40
			isHungry()	piranha.setHunger(29) piranha.isHungry() piranha.setHunger(30) piranha.isHungry() piranha.setHunger(31) piranha.isHungry()	true true false	true true false
8	Snail	SnailTest	setMoney	snail.SetMoney(new Node<> 10) snail.getMoney()	10	10
			getRadius	snail.getRadius()	30	30
			grabCoin	snail.setMoney(new Node<> 10) snail.getMoney() coin = new Coin(30, 570, 50) listOfCoin.getCount() snail.grabCoin(listOfCoin.get(0)) snail.getMoney()	10 1 60 0	10 1 60 0

				listOfCoin.getCount()		
9	Aquarium	AquariumTest	loadData	Membaca save.dat	25 1 300.0 550.0 0 1 400.0 300.0 0 0	25 1 300.0 550.0 0 1 400.0 300.0 0 0
			saveData	Menulis save.dat	25 1 300.0 550.0 0 1 400.0 300.0 0 0	25 1 300.0 550.0 0 1 400.0 300.0 0 0

7 Pengukuran Metriks Aplikasi

- Package: Default

Stats:
Total Classes: 9
Concrete Classes: 9
Abstract Classes: 0

Ca: 0
Ce: 5

A: 0
I: 1
D: 0

Abstract Classes:

Concrete Classes:
AquariumObjectTest
AquariumTest
CoinTest

FishFoodTest
GuppyTest
LinkedListTest
NodeTest
PiranhaTest
SnailTest

Depends Upon:
itb.akadquarium
java.io
java.lang
java.util
org.junit.jupiter.api

Used By:
Not used by any packages.

- Package: itb.akadquarium

Stats:
Total Classes: 11
Concrete Classes: 9
Abstract Classes: 2

Ca: 1
Ce: 8

A: 0.18
I: 0.89
D: 0.07

Abstract Classes:
itb.akadquarium.AquariumObject
itb.akadquarium.Fish

Concrete Classes:
itb.akadquarium.Aquarium
itb.akadquarium.Aquarium$1
itb.akadquarium.Coin
itb.akadquarium.FishFood
itb.akadquarium.Guppy
itb.akadquarium.LinkedList
itb.akadquarium.Node

itb.akadquarium.Piranha
itb.akadquarium.Snail

Depends Upon:
java.awt
java.awt.event
java.awt.image
java.io
java.lang
java.time
java.util
javax.swing

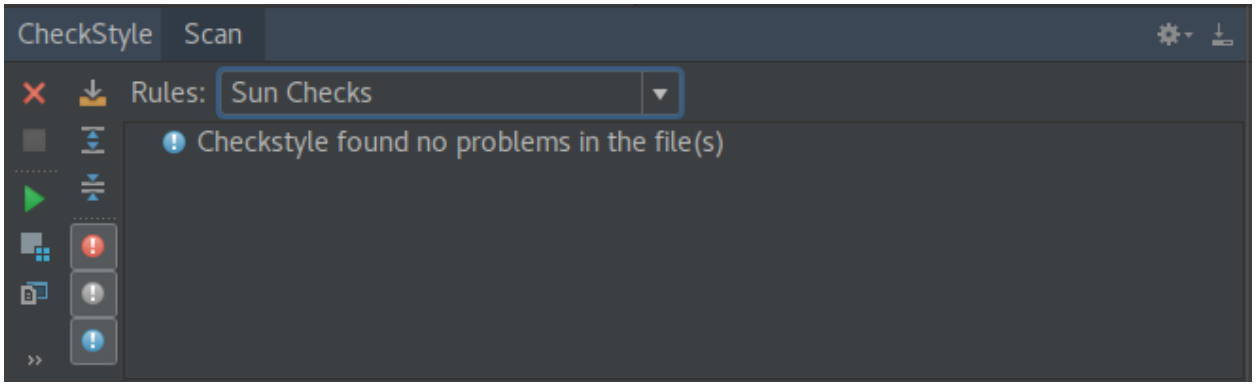
Used By:
Default

- Summary:

Name, Class Count, Abstract Class Count, Ca, Ce, A, I, D, V:

Default,9,0,0,5,0,1,0,1
itb.akadquarium,11,2,1,8,0.18,0.89,0.07,1
java.awt,0,0,1,0,0,0,1,1
java.awt.event,0,0,1,0,0,0,1,1
java.awt.image,0,0,1,0,0,0,1,1
java.io,0,0,2,0,0,0,1,1
java.lang,0,0,2,0,0,0,1,1
java.time,0,0,1,0,0,0,1,1
java.util,0,0,2,0,0,0,1,1
javax.swing,0,0,1,0,0,0,1,1
org.junit.jupiter.api,0,0,1,0,0,0,1,1

8 Pengukuran Kualitas Kode Aplikasi



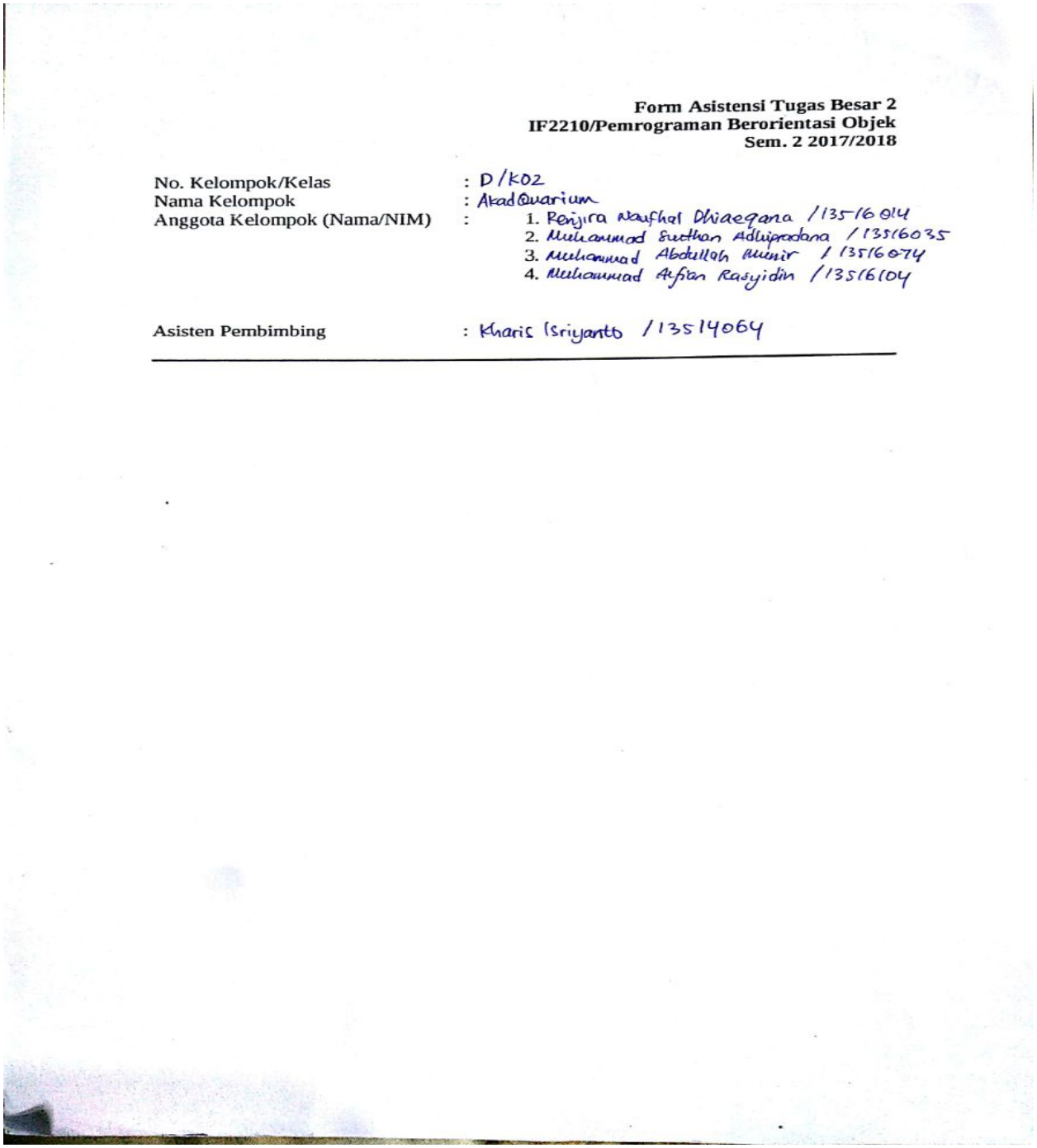
Gambar 2. Hasil Analisis Menggunakan *Checkstyle*

9 Pembagian Kerja dalam Kelompok

NIM	Uraian Pekerjaan
13516014	PJ CheckStyle, Guppy, Piranha, AquariumObject
13516035	PJ JUnit, Main, Aquarium, Snail, Main Menu, LinkedList
13516074	PJ Depend, Save and Load, Visual dan Audio, Node
13516104	PJ JavaDocs, Coin, Fish, FishFood, Laporan

10 Lampiran

10.1 Form Asistensi

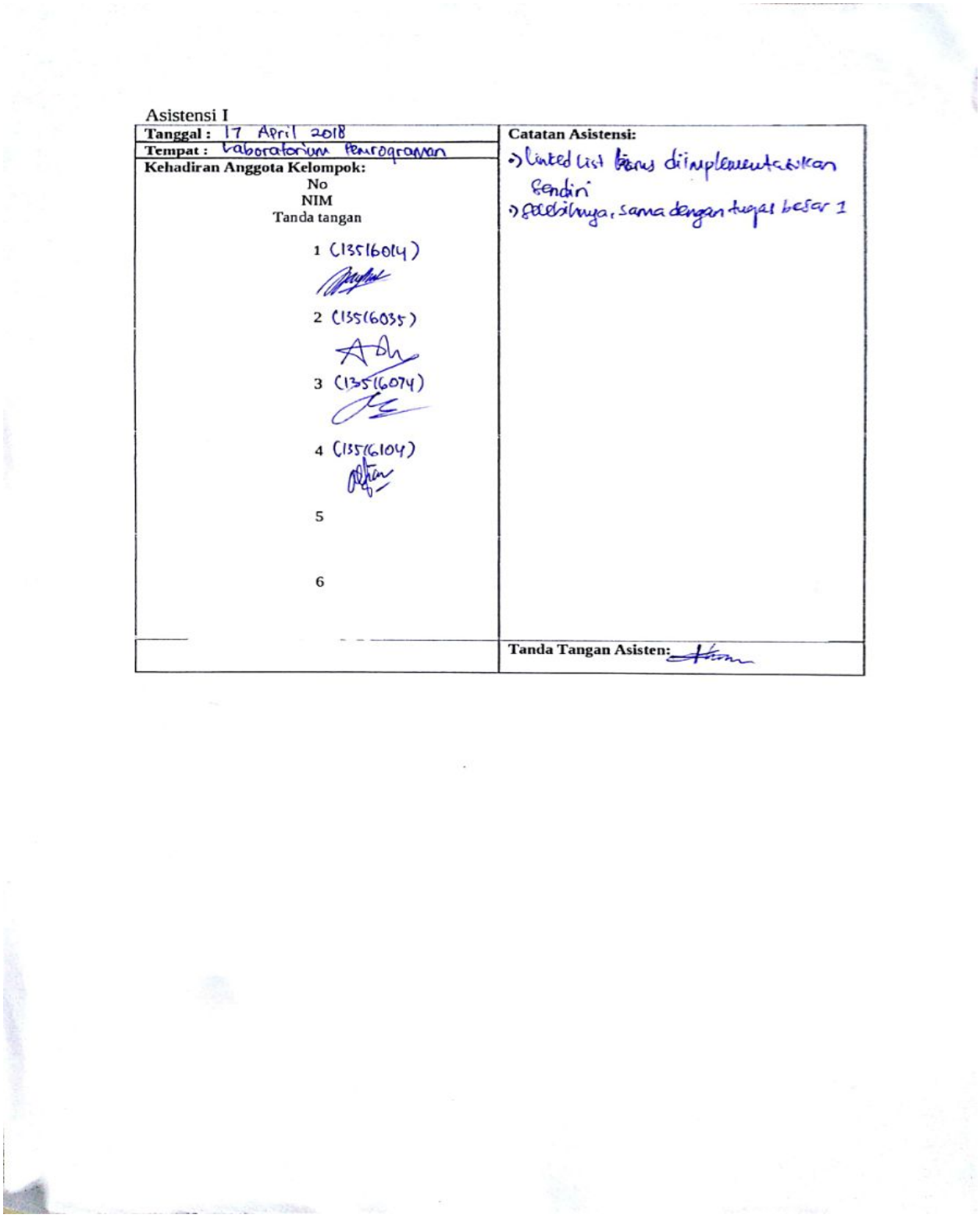


Form Asistensi Tugas Besar 2
IF2210/Pemrograman Berorientasi Objek
Sem. 2 2017/2018

No. Kelompok/Kelas : D/K02
Nama Kelompok : AkadQuarium
Anggota Kelompok (Nama/NIM) :
1. Renjira Naufal Dhiaegana /13516014
2. Muhammad Ruzhan Adliprodana /13516035
3. Muhammad Abdullah Munir /13516074
4. Muhammad Afsan Rasyidin /13516104

Asisten Pembimbing : Kharis Isriyanto /13514064

Gambar 3. Cover Form Asistensi



Asistensi I

Tanggal : 17 April 2018	**Catatan Asistensi:**
Tempat : Laboratorium Pemrograman	-> Linked list harus diimplementasikan sendiri
Kehadiran Anggota Kelompok:	-> Selebihnya, sama dengan tugas besar 1
No / NIM / Tanda tangan	
1 (13516014)	
2 (13516035)	
3 (13516074)	
4 (13516104)	
5	
6	
	Tanda Tangan Asisten:

Gambar 4. Form Asistensi 1

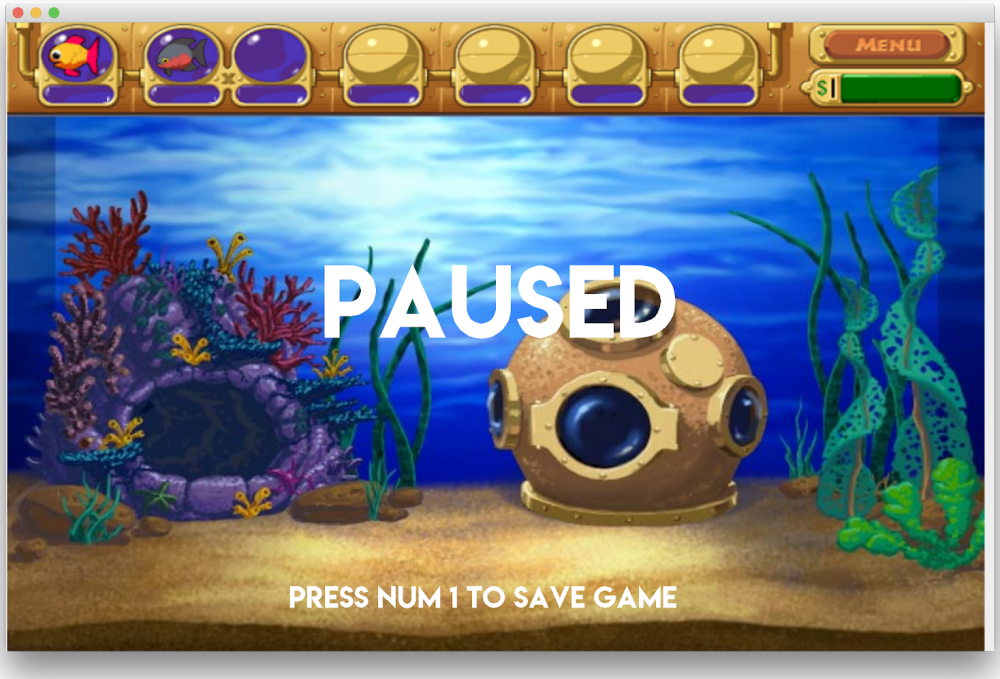
10.2 Log Activity Anggota Kelompok

Hari/Tanggal	13516014	13516035	13516074	13516104
17/4/2018	Diskusi pembagian tugas dan desain Asistensi	Diskusi pembagian tugas dan desain Asistensi	Diskusi pembagian tugas dan desain Asistensi	Diskusi pembagian tugas dan desain Asistensi
22/4/2018	Membuat program kelas-kelas yang ditugasi	Membuat program kelas-kelas yang ditugasi	Membuat program kelas-kelas yang ditugasi	Membuat program kelas-kelas yang ditugasi
23/4/2018	*Parallel programming Test* dan *Debugging*	*Parallel programming Test* dan *Debugging*	*Parallel programming Test* dan *Debugging*	*Parallel programming Test* dan *Debugging*
24/4/2018	Membuat JUnit di kelas yang ditugasi	Membuat JUnit di kelas yang ditugasi	Membuat JUnit di kelas yang ditugasi	Membuat JUnit di kelas yang ditugasi
25/4/2018	Finalisasi Checkstyle	Finalisasi Junit	Finalisasi *test* dan *debugging* Analisis JDepend	Finalisasi JavaDocs Membuat laporan

10.3 Screenshot Program

Gambar 5. Tampilan Beranda Permainan

Gambar 6. Tampilan Permainan Berjalan



Gambar 7. Tampilan *Pause*