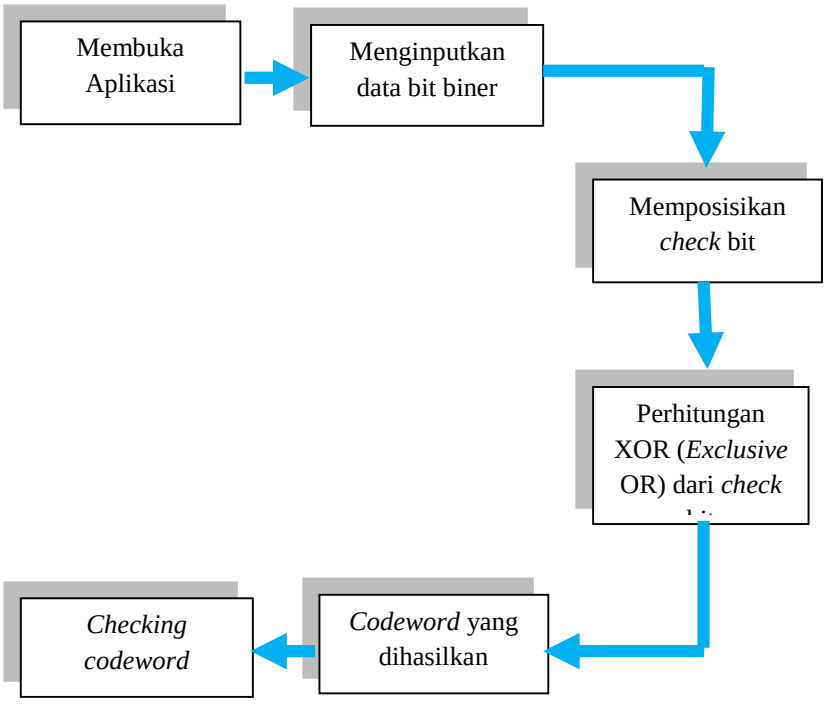
BAB III
PERANCANGAN APLIKASI

Pemodelan sistem Tugas Akhir aplikasi simulasi *error correction* pada komunikasi data menggunakan Kode Hamming berbasis Android yang akan dijalankan pada *smartphone* berbasis Android dengan versi Android minimal 2.3.

3.1 Blok Diagram Sistem

Dalam Tugas Akhir ini aplikasi simulasi *error correction* menggunakan Kode Hamming terbagi menjadi enam bagian utama, membuka aplikasi menggunakan *smartphone* berbasis Android, kemudian menginputkan data bit biner yang akan disimulasikan, memposisikan *check bit* menggunakan formula Kode Hamming, perhitungan XOR dari *check bit* untuk menghasilkan *codeword*, kemudian melakukan pengecekan *codeword* akan menampilkan hasil *codeword* yang benar. Seperti tampak dalam blok diagram yang dapat dilihat pada gambar 3.1.



Gambar 3.1. Blok Diagam Sistem

Gambar 3.1. Blok Diagram Sistem menunjukan alur kerja dalam menjalankan simulasi *error correction*, dimulai dengan meng-*input*kan data biner hingga menghasilkan *codeword* yang benar

1. Membuka Aplikasi 33

   Pada Tugas Akhir ini Aplikasi dijalankan pada *smartphone* berbasis Android. *Smartphone* yang digunakan adalah Samsung Galaxy Mini dengan versi Android 2.3. Aplikasi simulasi *error correction* dapat dijalankan jika sudah mengunduh di Play Store Android.

2. Menginputkan Data Bit Biner

   Dalam menjalankan Apikasi simulasi *error correction* menggunakan input bit biner, karena konsep Kode Hamming menggunakan formula penyisipan *check bit.* Bit biner yang diinputkan merupakan bit yang akan dilakukan perhitungan menggunakan Kode Hamming. Panjang data bit biner yang diinputkan maksimal 8 bit.

3. Memposisikan *Check Bit*

   Pada Tugas Akhir ini memposisikan *check bit* ke dalam panjang data bit biner yang diinputkan digunakan sebagai perhitungan untuk mendapatkan posisi bit yang bernilai 1. Posisi *check bit* disesuaikan dengan panjang data bit yang diinputkan. Sehingga dihasilkan panjang data *biner* yang baru.

4. Perhitungan XOR dari *Check Bit*

   Pada Tugas Akhir ini perhitungan gerbang logika XOR digunakan untuk menghitung dari nilai *bit* biner bernialai 1 yang terdapat pada *check bit* agar dihasilkan *codeword.*

5. *Codeword* yang dihasilkan

   Pada Tugas Akhir ini *codeword* yang dihasilkan digunakan sebagai parameter terjadinya *error* atau tidak pada *bit* yang diinputkan.

6. *Checking Codeword*

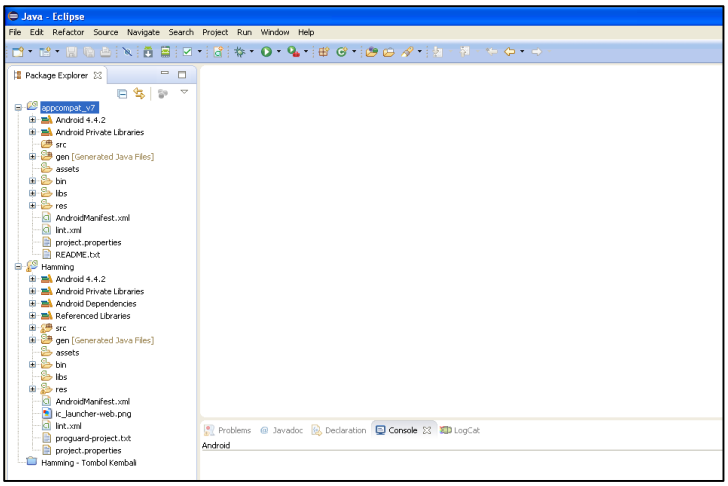
   Pada Tugas Akhir ini *checking codeword* digunakan untuk mengetahui bahwa *codeword* yang dihasilkan benar.

3.2 Analisa Kebutuhan

Dalam pembuatan aplikasi simulasi *error correction* menggunakan Kode Hamming pada komunikasi data berbasis Android dibutuhkan sebuah *software* dan *hardware* agar aplikasi dapat diwujudkan sesuai yang diharapkan. Adapun *software* dan *hardware* yang diperlukan dalam pembuatan aplikasi simulasi *error correction.*

3.2.1. Analisa Kebutuhan *Software*

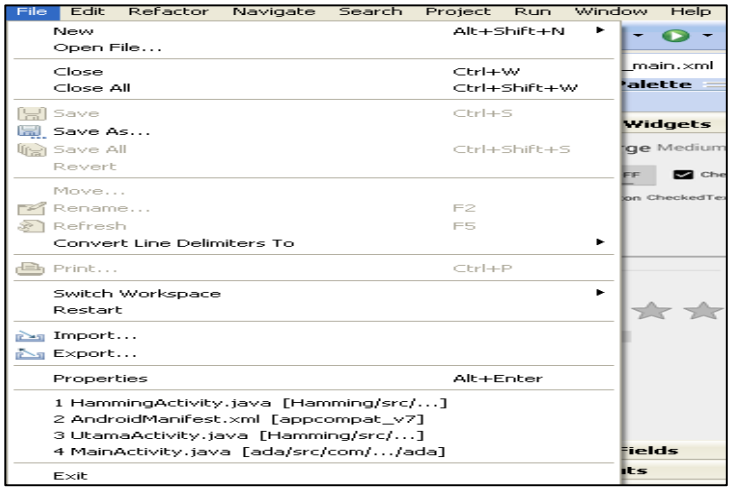
Software yang dibutuhkan dalam membangun aplikasi *error correction* yakni *software Eclipse. Software* ini digunakan untuk menuliskan *script* dalam membangun aplikasi simulasi *error correction,* dengan sifatnya yang *open source* dan gratis penggunaan *software* ini banyak digunakan dalam membangun suatu program aplikasi. Pada *software Eclipse* terdapat berbagai menu pilihan mulai dari *menu file, menu edit, menu refactor, menu source, menu navigate, menu search, menu project, menu run, menu window* dan *menu help.* Serta terdapat *icon menu* lainnya yang memiliki fungsi masing-masing. Dapat dilihat pada gambar 3.2.



Gambar 3.2. Tampilan *Software Eclipse*

Gambar 3.2. menunjukan tampilan menu yang terdapat pada *software Eclipse* yang umum digunakan antara lain *menu file* dan *menu edit.*

1. *Menu file* berisi beberapa *submenu* yang sering digunakan,dapat dilihat pada gambar 3.3 antara lain:
   a. *New*, digunakan untuk membuat *file project* baru.
   b. *Open file*, digunakan untuk membuka *file project* yang telah dibangun.
   c. *Save*, digunakan untuk menyimpan *file project* yang telah dibuat.
   d. *Rename*, digunakan untuk mengganti nama *file project.*
   e. *Import*, digunakan untuk mengambil *file project* yang berada di*folder* lain.
   f. *Export*, digunakan untuk meng-*export file project* ke tempat penyimpanan yang baru.
   g. *Properties*, digunakan untuk melihat informasi *file project* antara lain, kapasitas, tempat penyimpanan, tipe data dan *attribute.*



Gambar 3.3 Tampilan *Submenu* dari *Menu File*

Gambar 3.3 di atas menunjukan *submenu* dari *menu file* yang digunakan dalam merancang sebuah program aplikasi.

5. *Menu Edit,* pada *menu* ini berisi beberapa *submenu* yang sering digunakan dalam pembuatan program aplikasi antara lain dapat dilihat pada gambar 3.4.
   a. *Undo*, perintah ini digunakan untuk membatalkan perintah sebelumnya

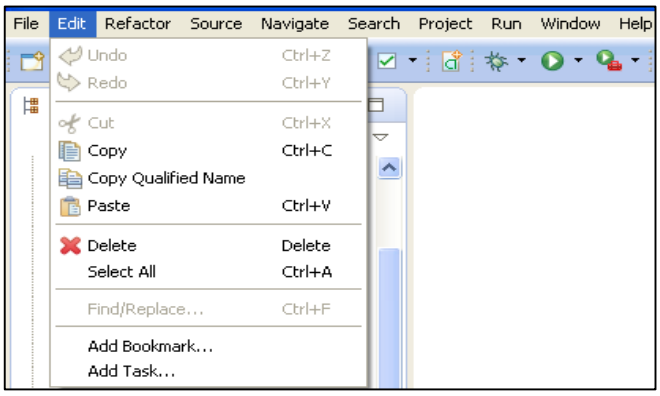
b. *Redo*, perintah ini digunakan untuk membatalkan perintah sesudahya

c. *Cut*

d. *Copy*, perintah ini digunakan untuk menduplikasi *file project*

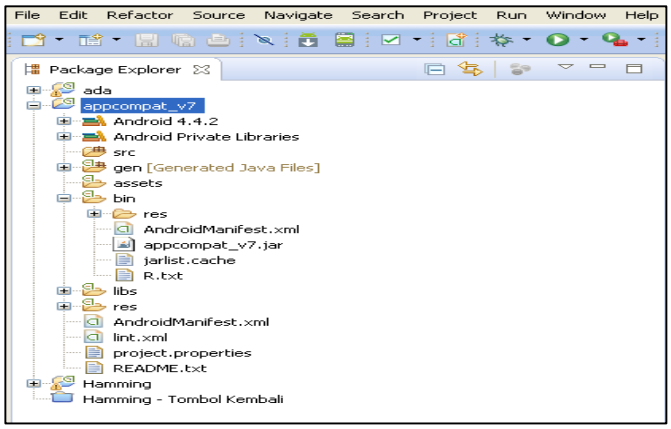
e. *Paste*

f. *Delete*, perintah ini digunakan untuk menghapus *file poject*



Gambar 3.4 Tampilan *Submenu* dari *Menu Edit*

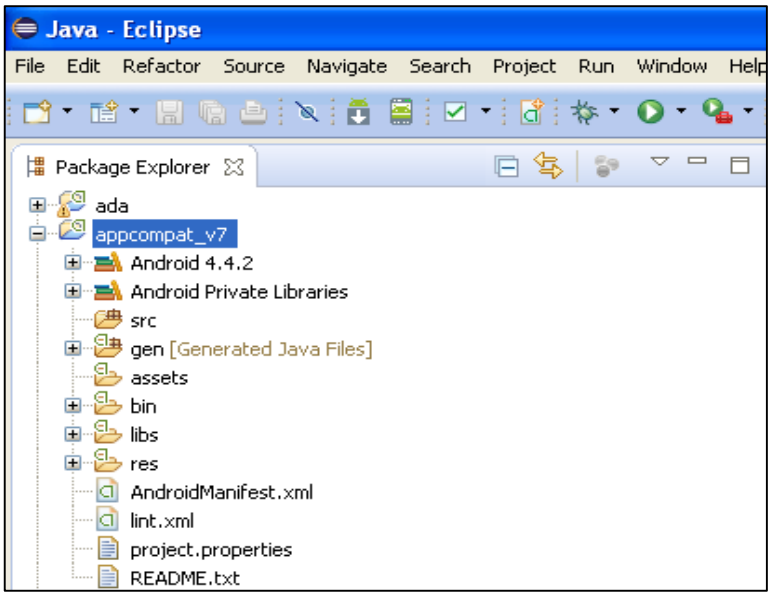
Gambar 3.4. menunjukan beberapa *submenu* yang terdapat pada *menu edit.* Pada umumnya fungsi dari perintah yang terdapat pada *menu edit* sama dengan perintah dalam membuat dokumen pada *Microsoft office.* Pada lembar kerja *Eclipse* terdapat *Package Explorer* yang berisi *file-file project* di dalam *Eclipse. File project* ini berupa program aplikasi yang telah dibangun dan dapat dimodifikasi karena di dalamnya terdapat *menu-menu* yang digunakan dalam membangun perangkat lunak yang dapat dilihat pada gambar 3.5.



Gambar 3.5. Tampilan *Package Explorer*

Gambar 3.5. menunjukan tampilan *package explorer*, di dalam *package explorer* terdapat *folder default* dari *Eclipse* yaitu *Appcompact_7*. *Appcompact_7* merupakan sebuah *folder library Eclipse*. Di dalam *folder appcompact_7* terdapat beberapa *subfolder* yang dapat dilihat pada gambar 3.6.antara lain:

1. Andorid 4.4.2
2. Android *Private Libraries*, berisi *folder src*, *src* merupakan *folder* yang berisi *source code* serta *package* yang digunakan dalam pembuatan aplikasi.
3. *Gen (Generated Java Files)*, merupakan *folder* yang berisi *source code* dari Java.
4. *Bin*, merupakan *folder* yang berisi *file* apk (*Android Package). File apk* ini adalah yang nantinya akan diinstal pada *smartphone* Android.
5. *Res*, atau *resource* merupakan *folder* yang berisi dalam menyusun aplikasi, antara lain *drawable hdpi, drawable ldpi, drawable mdpi, drawable xhdpi, drawable xxhdpi, layout, menu, values, values v11, values v14, values w820dp.*



Gambar 3.6. Tampilan *folder Appcompact_7*

Gambar 3.6. menunjukan tampilan dari *folder Appcompact_7* yang berisi *file-file library.*

3.2.2. Analisa Kebutuhan *Hardware*

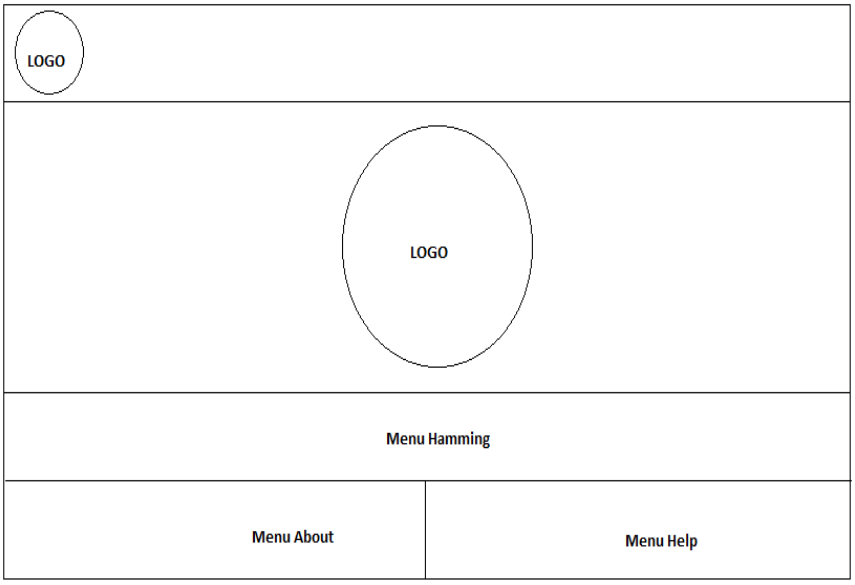
Software Eclipse yang digunakan dalam membangun aplikasi simulasi *error correction* pada komunikasi data menggunakan Kode Hamming memerlukan spesifikasi minimum *hardware* yakni minimal *memory 2 gb.* Sedangkan Laptop yang digunakan untuk membangun aplikasi simulasi *error correction* ini memiliki spesifikasi sebagai berikut:

1. Laptop Axio, *Processor* Intel Core 2 *Duo*
6. RAM 4 Gb
7. Windows XP *Professional*
8. *Smartphone* Android Samsung Galaxy Mini dengan sistem operasi Android versi 2.3.

3.3 Perancangan Aplikasi

3.3.1. Rancangan *Layout* Aplikasi

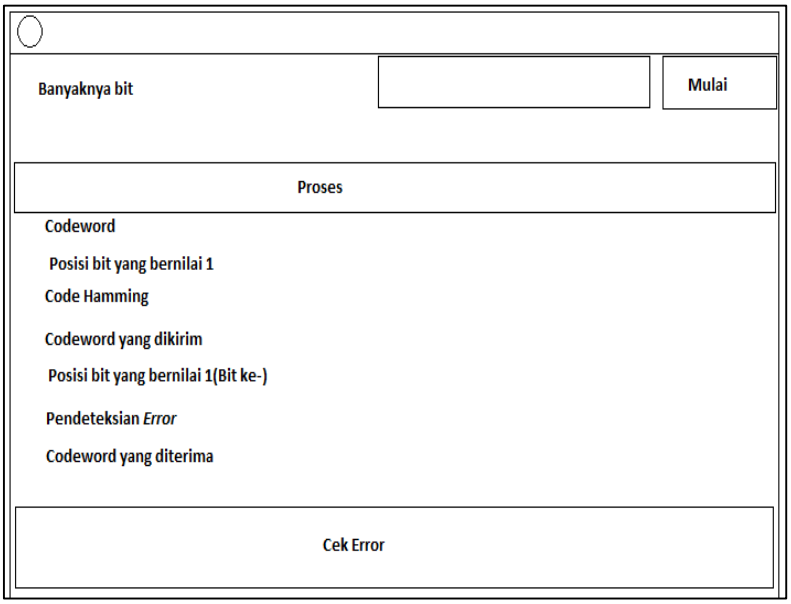
Pada perancangan aplikasi simulasi *error correction* menggunakan Kode Hamming terdiri dari membangun *layout* dan bahasa pemrograman yang digunakan. Untuk Perancangan *layout* dari aplikasi simulasi *error correction* ini dapat dilihat pada gambar 3.7.



Gambar 3.7. Tampilan *Menu* Utama

Gambar 3.7. menunjukan isi dari tampilan *menu* utama, yaitu terdapat logo aplikasi simulasi dan tiga *menu* utama yaitu *menu hamming*, *menu about* dan *menu help*, pada tampilan *menu* utama dari aplikasi simulasi *error correction* mempunyai tiga *menu* utama yakni:

1. *Menu* Hamming, merupakan *menu* yang digunakan untuk melakukan perhitungan simulasi cek *error* dapat dilihat pada gambar 3.9 di bawah ini terdiri dari beberapa kolom antara lain:
   a. Banyaknya *Bit*, kolom ini berisi banyaknya data *biner* yang disimulasikan
   b. Data *Bit*, kolom ini berisi data *bit biner* yang dikirimkan
   c. *Codeword*, kolom ini berisi *check bit* yang disisipkan pada data *bit*
   d. Posisi bit yang bernilai 1 (Bit ke-), berisi letak *bit biner* yang bernilai 1
   e. *Code Hamming*, kolom ini berisi hasil perhitungan dari gerbang logika XOR dari *bit biner* yang bernilai 1
   f. *Codeword* yang dikirim, kolom ini berisi data bit dikirimkan yang disisipkan *bit biner* hasil pada kolom *code hamming*
   g. Posisi *bit* yang bernilai 1 (Bit ke-)
   h. Pendeteksian *error*, hasil yang diperoleh dari perhitungan

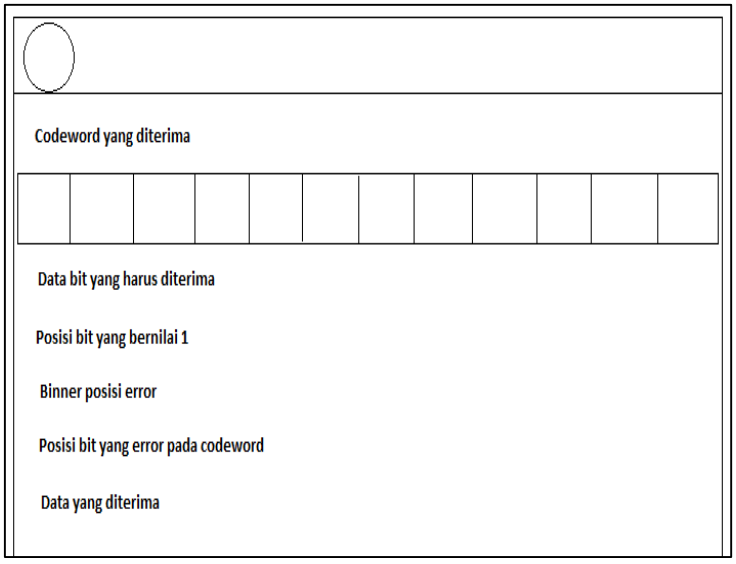


Gambar 3.8. Tampilan *Menu Hamming*

40Gambar 3.8. menunjukan desain rancangan *menu hamming*, pada *menu* ini terdapat *submenu* cek *error*. *Submenu* ini menampilkan *activity* yang menampilkan kolom-kolom yang menjelaskan proses dari cek *error* dari data bit yang disimulasikan. Berbeda dengan *menu Hamming*, *user* hanya dapat menentukan letak *bit error* yang akan disimulasikan.

Dalam menentukan letak cek *error* terdapat pada kolom *codeword* yang diterima. Jumlah *bit* kesalahan maksimal yang dapat diinputkan adalah satu *bit error*. Karena Kode Hamming merupakan metode koreksi dengan *single bit error*. Tampilan *submenu* cek *error* dapat dilihat pada gambar 3.9. dibawah ini yang terdiri dari:

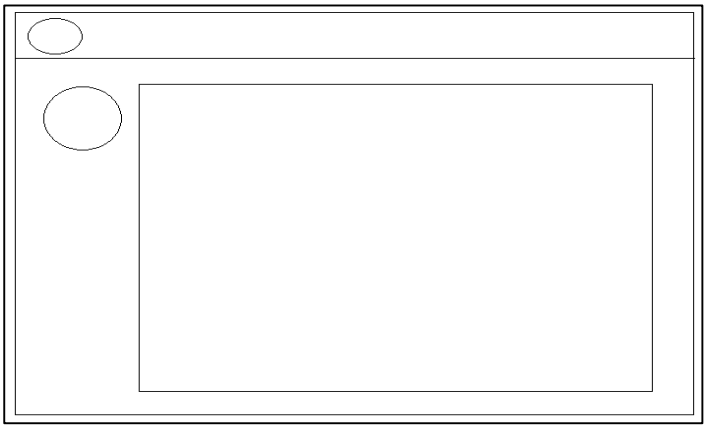
a. *Codeword* yang diterima
b. Data bit yang harus diterima
c. Posisi *bit* yang bernilai 1
d. Binner posisi *error*
e. Posisi *bit* yang *error* pada *codeword* (Bit ke-)
f. Data yang diterima



Gambar 3.9.Tampilan *submenu* cek *error*

Gambar 3.9. menunjukan desain tampilan *submenu* dari cek *error*, terdapat beberapa kolom yang menjelaskan proses perhitungan koreksi kesalahan.

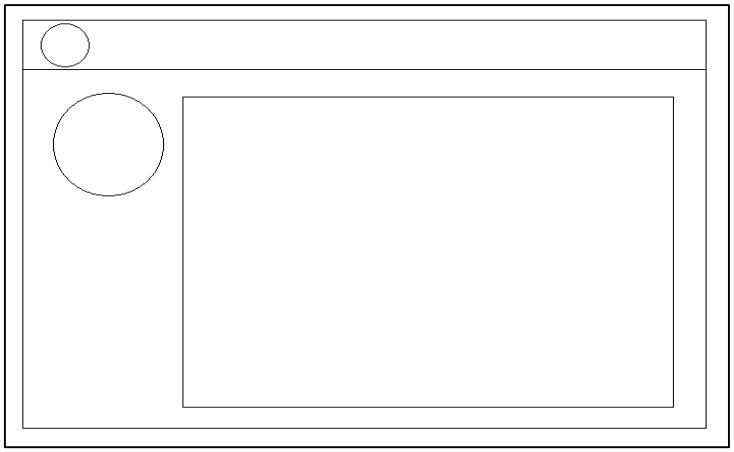
3. *Menu About*, merupakan *menu* yang berisi mengenai data diri pembuat aplikasi simulasi *error correction* seperti nama pembuat dan nama kampus, serta berisi sedikit penjelasan mengenai metode koreksi kesalahan menggunakan Kode Hamming. Tampilan *menu about* dapat dilihat pada gambar 3.10.



Gambar 3.10. Tampilan *Menu About*

Gambar 3.10. di atas menunjukan tampilan *menu about*, *menu* ini berfungsi untuk memberikan informasi mengenai pembuat aplikasi dan gambaran mengenai Kode Hamming

4. Menu Help, menu ini merupakan *menu* yang dapat digunakan pembaca untuk memahami bagaimana tata cara menjalankan aplikasi simulasi *error correction* menggunakan Kode Hamming. Tampilan *menu help* dapat dilihat pada gambar 3.11. Gambar 3.11. menunjukan tampilan *menu help, menu* ini berisi langkah – langkah dalam menggunakan aplikasi simulasi *error correction.*



Gambar 3.11 Tampilan *Menu Help*

3.3.2. *Source Code* Aplikasi Simulasi

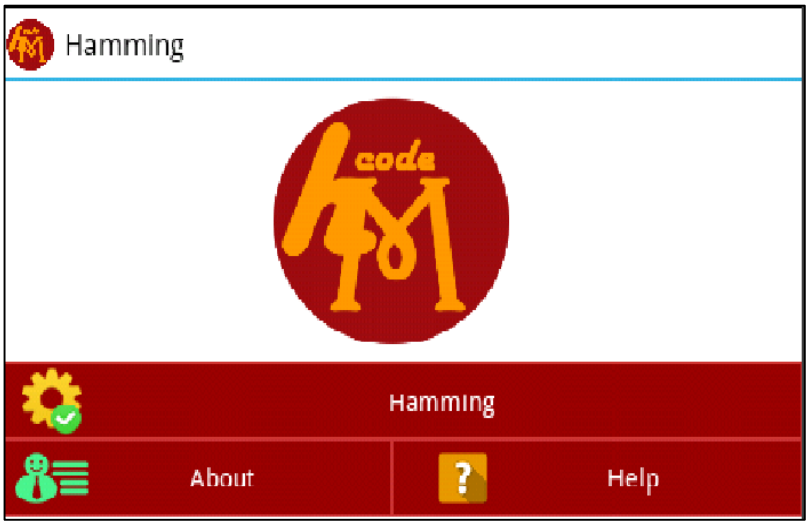
Dalam membangun aplikasi simulasi *error correction* menggunakan bahasa pemrograman Java, karena bahasa pemrograman Java memiliki konsep berorientasi objek. Adapun *source code* yang dituliskan dapat dilihat pada gambar 3.12.

```
package com.dela.hamming;

import com.dela.hamming.R;
import android.support.v7.app.ActionBarActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;

public class UtamaActivity extends ActionBarActivity
{
       private Button bnUtamaHamming, bnUtamaAbout,
bnUtamaHelp;
@Override
public void onClick(View v) {
                              // untuk menampilkan
tampilan Utama
Intent vwIntent = new Intent(UtamaActivity.this,
HammingActivity.class);
                              startActivity(vwIntent);
```

Gambar 3.12. *Source Code* Tampilan *Menu* Utama

Gambar 3.12. *Source Code* Menampilkan Tampilan *Menu* Utama

Berdasarkan *source code* pada gambar 3.12. maka output yang dihasilkan dapat dilihat pada gambar 3.13.



Gambar 3.13. Tampilan *Menu* Utama

Untuk membuat tombol pada Menu Utama menggunakan *field button*. Dalam membuat tombol dapat dimodifikasi dengan memberikan warna, teks dan *icon* yang dapat dilihat pada gambar 3.14.

```
<Button
            android:id="@+id/bnUtamaHamming"
            android:layout_width="0dp"

android:layout_height="wrap_content"
            android:layout_weight="1"

android:background="@drawable/button_flat"

android:drawableLeft="@drawable/process64"
            android:drawablePadding="15dp"
            android:gravity="center"
            android:paddingLeft="20dp"
            android:paddingRight="20dp"
            android:paddingStart="20dp"
            android:text="Hamming"
            android:textColor="#fff" />
```

Gambar 3.14. *Source Code* Membuat Tombol

Untuk menampilkan agar menu Hamming dapat di-klik *source code* yang dituliskan dapat dilihat pada gambar 3.15.

```
            @Override
      protected void onCreate(Bundle
savedInstanceState) {
            super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_utama);

            bnUtamaHamming = (Button)
findViewById(R.id.bnUtamaHamming);


            bnUtamaHamming.setOnClickListener(new
            OnClickListener() {
```

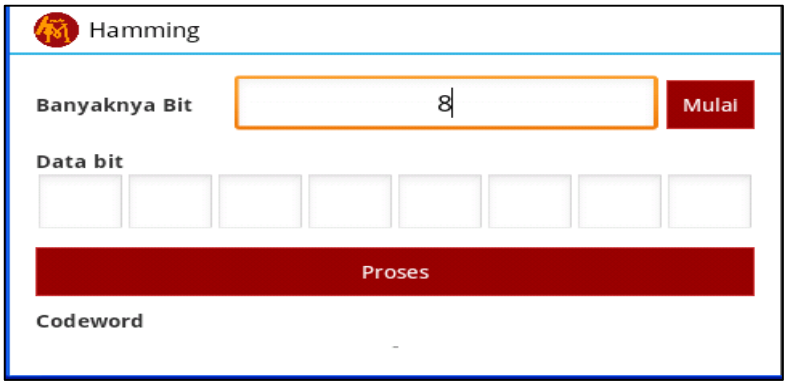
Gambar 3.15. *Source Code OnClickListener*

Untuk membuat formula *layout* dari *menu* Hamming dapat dilihat pada gambar 3.16.

```
package com.dela.hamming;

public class Hamming {
	private int iMax = 9, iOutMax = 13,
nHamming = 0;//memunculkan banyaknya bit
	private int iListCodeWordBitSatu[] = new
int[iOutMax];
	private String sInputData = "";

	private String CodeWord = "",
CodeWordSatu = "", CodeHamming = "",
			CodeWordKirim = "";

	public Hamming() {
		ProsesHamming();
	}

	public Hamming(String data) {
		this.sInputData = data;

		ProsesHamming();
	}

	private void ProsesHamming() {
		setCodeWord();
		setCodeWordSatu();
		setCodeWordKirim();
	}

	/*	}
```

Gambar 3.16. *Source Code* Tampilan Input Banyaknya Bit

Berdasarkan gambar 3.16. dihasilkan tampilan yang dapat dilihat pada gambar 3.17.



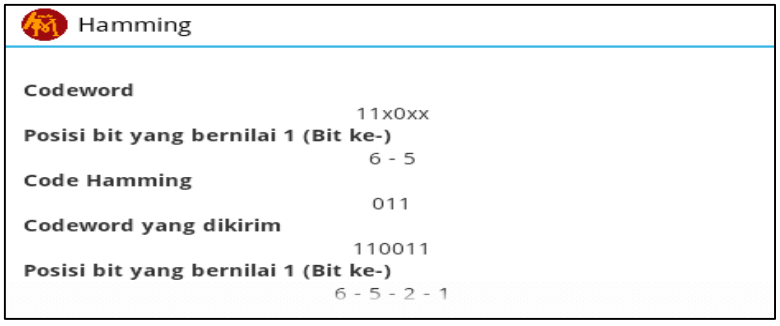
Gambar 3.17. *Input* Banyaknya Bit

Untuk menampilkan kotak dialog *codeword* sehingga dihasilkan *check bit* yang disisipkan pada panjang data *input* dapat dilihat pada gambar 3.18.

```
private void setCodeWord() {
		int nData = sInputData.length();
		if (nData > 8) {
			nHamming = 5;
		} else if (nData > 4) {
			nHamming = 4;
		} else if (nData >= 2) {
			nHamming = 3;

		}
		int nCodeword = nData + nHamming;
		String sCodeWord = "";
		for (int i = 1; i <= nCodeword; i++) {
			if (i == 1 || i == 2 || i == 4 || i
== 8 || i == 16) {
				sCodeWord = "x" + sCodeWord;
			} else {
				sCodeWord =
sInputData.charAt(--nData) + sCodeWord;
			}
		}
		this.CodeWord = sCodeWord;
	}

	public String getCodeWord() {
		return CodeWord;
```

Gambar 3.18. *Source Code* Kolom *Codeword*

Berdasarkan *source code* yang ditampilkan pada gambar 3.18. menghasilkan kolom *codeword* dengan posisi *check bit* yang ditandai tanda x, dapat dilihat pada gambar 3.19.



Gambar 3.19. Tampilan *Input* Kolom *Codeword*