

## 1. Game Screen Design – Shooting Game

1. Create a new project, name the project and the class at “*Shooting*” and “*MyClass*”.
2. Put the airplane and life images (PNG format) into the “res” folder (C:\WTK22\apps\Shooting\res).



3. Create two java files named “*MyClass.java*” and “*ImageCanvas.java*” in the “src” folder (C:\WTK22\apps\Shooting\src).

### MyClass.java

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*****
/* Sample Shooting Game.
*****/
public class MyClass extends MIDlet implements CommandListener {
    // Define the GUI components
    private ImageCanvas MyCanvas;      // Canvas
    private Command cmdExit;           // Exit Button
    private Command cmdStart;          // Start Button

    // Define the no-argument constructor
    public MyClass() {
        // Define the Exit Button
        cmdExit = new Command("Exit", Command.EXIT, 0);

        // Define the Start Button
        cmdStart = new Command("Start", Command.SCREEN, 1);

        // Define the canvas
        MyCanvas = new ImageCanvas();
    }

    // Called by application manager to start the MIDlet
    public void startApp() {
        // Add the Command button and the Command Listener
        MyCanvas.addCommand(cmdExit);
        MyCanvas.addCommand(cmdStart);
        MyCanvas.setCommandListener(this);

        // Set the current display of the midlet to the Canvas
        Display.getDisplay(this).setCurrent(MyCanvas);
    }

    // PauseApp is used to suspend background activities and release resources
    // on the device when the midlet is not active.
    public void pauseApp() {
    }

    // DestroyApp is used to stop background activities and release
```

```

// resources on the device when the midlet is at the end of its life cycle.
public void destroyApp(boolean unconditional) {
}

// Implement the event handling method defined in the CommandListener interface.
public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
        destroyApp(true);
        notifyDestroyed();
    }
}
}

```

### ImageCanvas.java

```

// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*-----*/
/* Canvas for graphic display */
/*-----*/
public class ImageCanvas extends Canvas {
    private Image imgPlane; // Graphic for Plane
    private Image imgLife; // Graphic for Life
    private int PlaneY = 70; // Y-coordinate for Air plane
    private int i; // Looping variables
    private int CurrentLife = 3; // Current Life
    private int score = 0; // Display the Score

    public ImageCanvas() {
        // Load the image from resource folder
        try {
            imgPlane = Image.createImage("/plane.png"); // Air Plane
            imgLife = Image.createImage("/life.png"); // Life
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    // Paint the graphic on the screen
    public void paint(Graphics g) {
        try {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Draw the ground
            g.setColor(0, 255, 255);
            g.fillRect(0, 150, 180, 180);

            // Draw the Air Plane
            g.drawImage(imgPlane, 0, PlaneY, g.TOP|g.LEFT);

            // Draw the score
            g.setColor(0, 0, 0);
            g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

            // Display the number of life
            for (i=1; i<=CurrentLife; i++) {
                g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}

```

4. Compile and execute the MIDlet, you can see the draft screen for game.



## 2. Add the Opening

1. Open the project “*Shooting*” and put the opening image (PNG format) into the resource folder (C:\WTK22\apps\Shooting\res).



2. Modify the source file “*MyClass.java*” and “*ImageCanvas.java*” in the “src” folder (C:\WTK22\apps\Shooting\src).

### MyClass.java

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*****
/* Sample Shooting Game.
*****/

public class MyClass extends MIDlet implements CommandListener {
    // Define the GUI components
    private ImageCanvas MyCanvas;           // Canvas
    private Command cmdExit;                // Exit Button
    private Command cmdStart;              // Start Button
    private Thread MyThread;                // Thread

    // Define the no-argument constructor
    public MyClass() {
        // Define the Exit Button
        cmdExit = new Command("Exit", Command.EXIT, 0);

        // Define the Start Button
        cmdStart = new Command("Start", Command.SCREEN, 1);

        // Define the canvas
        MyCanvas = new ImageCanvas();
    }

    // Called by application manager to start the MIDlet
    public void startApp() {
        // Add the Command button and the Command Listener
        MyCanvas.addCommand(cmdExit);
        MyCanvas.addCommand(cmdStart);
        MyCanvas.setCommandListener(this);

        // Set the current display of the midlet to the Canvas
        Display.getDisplay(this).setCurrent(MyCanvas);
    }
}
```

```

}

// PauseApp is used to suspend background activities and release resources
// on the device when the midlet is not active.
public void pauseApp() {
}

// DestroyApp is used to stop background activities and release
// resources on the device when the midlet is at the end of its life cycle.
public void destroyApp(boolean unconditional) {
}

// Implement the event handling method defined in the CommandListener interface.
public void commandAction(Command c, Displayable s) {
    if (c == cmdStart) {
        // Start the thread only one time
        if (MyThread == null) {
            // Define the thread
            MyThread = new Thread(MyCanvas);

            // Start the thread
            MyThread.start();
        }
    } else if (c == cmdExit) {
        destroyApp(true);
        notifyDestroyed();
    }
}
}

```

### ImageCanvas.java

```

// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*-----*/
/* Canvas for graphic display */
/*-----*/
public class ImageCanvas extends Canvas implements Runnable {
    private Image imgPlane; // Graphic for Plane
    private Image imgLife; // Graphic for Life
    private Image imgOpening; // Graphic for Opening
    private int PlaneY = 70; // Y-coordinate for Air plane
    private int i; // Looping variables
    private int CurrentLife = 3; // Current Life
    private int score = 0; // Display the Score
    private int ScreenMode = 0; // Game Screen Status
    private boolean ExitFlag = false; // The Flag to Exit

    // Define the game status
    final int GAME_OPEN = 0, // Opening
            GAME_PLAY = 1, // Playing
            GAME_OVER = 9; // Game Over

    public ImageCanvas() {
        // Load the image from resource folder
        try {
            imgPlane = Image.createImage("/plane.png"); // Air Plane
            imgLife = Image.createImage("/life.png"); // Life
            imgOpening = Image.createImage("/opening.png"); // Opening
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    // Execute the Thread
    public void run() {
        try {
            // Start the game
            ScreenMode = GAME_PLAY;

```

```

// Continuous to repaint the screen until the ExitFlag = True
while (!ExitFlag) {
    repaint(); // Call paint() to repaint the screen
    Thread.sleep(100); // Sleep for 0.1 second
}
} catch (Exception e) {
    System.out.println(e.getMessage());
}
}

// Paint the graphic on the screen
public void paint(Graphics g) {
    try {
        if (ScreenMode == GAME_OPEN) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Display the Opening
            g.drawImage(imgOpening, 0, 0, g.TOP/g.LEFT);
        } else if (ScreenMode == GAME_PLAY) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Draw the ground
            g.setColor(0, 255, 255);
            g.fillRect(0, 150, 180, 180);

            // Draw the Air Plane
            g.drawImage(imgPlane, 0, PlaneY, g.TOP/g.LEFT);

            // Draw the score
            g.setColor(0, 0, 0);
            g.drawString("Score:" + score, 0, 0, g.TOP/g.LEFT);

            // Display the number of life
            for (i=1; i<=CurrentLife; i++) {
                g.drawImage(imgLife, 110+i*18, 0, g.TOP/g.LEFT);
            }
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
}
}

```

3. Compile and execute the program, then you can see the opening screen, you can press **[Start]** to enter the game play screen.



### 3. Airplane Movement

1. Open the project “*Shooting*” and modify the source file “*ImageCanvas.java*” in the “*src*” folder (C:\WTK22\apps\Shooting\src).

#### ImageCanvas.java

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*-----*/
/* Canvas for graphic display */
/*-----*/
public class ImageCanvas extends Canvas implements Runnable {
    private Image imgPlane; // Graphic for Plane
    private Image imgLife; // Graphic for Life
    private Image imgOpening; // Graphic for Opening
    private int PlaneY = 70; // Y-coordinate for Air plane
    private int i; // Looping variables
    private int CurrentLife = 3; // Current Life
    private int score = 0; // Display the Score
    private int ScreenMode = 0; // Game Screen Status
    private boolean ExitFlag = false; // The Flag to Exit

    // Define the game status
    final int GAME_OPEN = 0, // Opening
            GAME_PLAY = 1, // Playing
            GAME_OVER = 9; // Game Over

    public ImageCanvas() {
        // Load the image from resource folder
        try {
            imgPlane = Image.createImage("/plane.png"); // Air Plane
            imgLife = Image.createImage("/life.png"); // Life
            imgOpening = Image.createImage("/opening.png"); // Opening
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    // Execute the Thread
    public void run() {
        try {
            // Start the game
            ScreenMode = GAME_PLAY;

            // Continuous to repaint the screen until the ExitFlag = True
            while (!ExitFlag) {
                repaint(); // Call paint() to repaint the screen
                Thread.sleep(100); // Sleep for 0.1 second
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    // KeyCode Event Handler to detecting game actions
    public void keyPressed(int keyCode) {
        // Move the air plane up when press [2]
        if (keyCode == KEY_NUM2) {
            PlaneY = 30;
        } // Move the air plane to middle when press [5]
        else if (keyCode == KEY_NUM5) {

```

```

    PlaneY = 70;
    // Move the air plane down when press [8]
    } else if (keyCode == KEY_NUM8) {
        PlaneY = 110;
    }
}

// Paint the graphic on the screen
public void paint(Graphics g) {
    try {
        if (ScreenMode == GAME_OPEN) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Display the Opening
            g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
        } else if (ScreenMode == GAME_PLAY) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Draw the ground
            g.setColor(0, 255, 255);
            g.fillRect(0, 150, 180, 180);

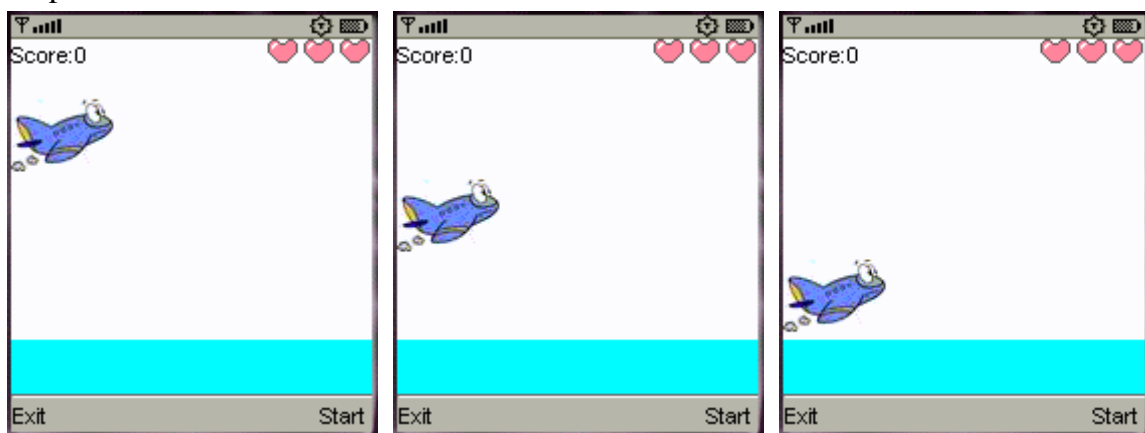
            // Draw the Air Plane
            g.drawImage(imgPlane, 0, PlaneY, g.TOP|g.LEFT);

            // Draw the score
            g.setColor(0, 0, 0);
            g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

            // Display the number of life
            for (i=1; i<=CurrentLife; i++) {
                g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
            }
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
}

```

2. Compile and execute the MIDlet, then press [2], [5] and [8] key on your mobile to control the air plane.



## 4. Create the Enemy (Missiles)

1. Open the project “*Shooting*” and put the missile image (PNG format) into the resource folder (C:\WTK22\apps\Shooting\res).



2. Modify the source file “*ImageCanvas.java*” in the “*src*” folder (C:\WTK22\apps\Shooting\src).

### ImageCanvas.java

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// import the random library
import java.util.Random;

/*-----*/
/* Canvas for graphic display */
/*-----*/
public class ImageCanvas extends Canvas implements Runnable {
    private Random rand = new Random(); // Random seek
    private Image imgPlane; // Graphic for Plane
    private Image imgLife; // Graphic for Life
    private Image imgOpening; // Graphic for Opening
    private Image imgMissile; // Graphic for Missile
    private int PlaneY = 70; // Y-coordinate for Air plane
    private int i; // Looping variables
    private int CurrentLife = 3; // Current Life
    private int score = 0; // Display the Score
    private int ScreenMode = 0; // Game Screen Status
    private boolean ExitFlag = false; // The Flag to Exit
    private int MissileNumber = 4; // Number of Missile

    // Missile position (X, Y) and speed V
    private int[] MissileX = new int[10];
    private int[] MissileY = new int[10];
    private int[] MissileV = new int[10];

    // Define the game status
    final int GAME_OPEN = 0, // Opening
            GAME_PLAY = 1, // Playing
            GAME_OVER = 9; // Game Over

    public ImageCanvas() {
        // Initialize the Missile position
        for (i=0; i<MissileNumber; i++) {
            MissileX[i] = 180; // Set the x-coordinate to 180
            MissileY[i] = 0; // Set the y-coordinate to 0
            MissileV[i] = 0; // Set the speed to 0
        }

        // Load the image from resource folder
        try {
            imgPlane = Image.createImage("/plane.png"); // Air Plane
            imgLife = Image.createImage("/life.png"); // Life
            imgOpening = Image.createImage("/opening.png"); // Opening
            imgMissile = Image.createImage("/missile.png"); // Missile
        }
    }
}
```

```

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

// Execute the Thread
public void run() {
    try {
        // Start the game
        ScreenMode = GAME_PLAY;

        // Continuous to repaint the screen until the ExitFlag = True
        while (!ExitFlag) {
            repaint(); // Call paint() to repaint the screen
            Thread.sleep(100); // Sleep for 0.1 second
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
    // Move the air plane up when press [2]
    if (keyCode == KEY_NUM2) {
        PlaneY = 30;
    } // Move the air plane middle when press [5]
    else if (keyCode == KEY_NUM5) {
        PlaneY = 70;
    } // Move the air plane down when press [8]
    else if (keyCode == KEY_NUM8) {
        PlaneY = 110;
    }
}

// Paint the graphic on the screen
public void paint(Graphics g) {
    try {
        if (ScreenMode == GAME_OPEN) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Display the Opening
            g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
        } else if (ScreenMode == GAME_PLAY) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Draw the ground
            g.setColor(0, 255, 255);
            g.fillRect(0, 150, 180, 180);

            // Draw the Air Plane
            g.drawImage(imgPlane, 0, PlaneY, g.TOP|g.LEFT);

            // Draw the missile
            for (i=0; i<MissileNumber; i++) {
                // Move the missile to attack air plane if speed > 0
                if (MissileV[i] > 0) {
                    if (MissileX[i] > 0) {
                        MissileX[i] = MissileX[i] - MissileV[i];
                    } else {
                        MissileV[i] = 0;
                    }
                } else if (Math.abs(rand.nextInt())%5 == 1) {
                    // Create an new missile
                    MissileX[i] = 180;
                    MissileY[i] = Math.abs(rand.nextInt())%3*40 + 30; // Random the location
                }
            }
        }
    }
}

```

```

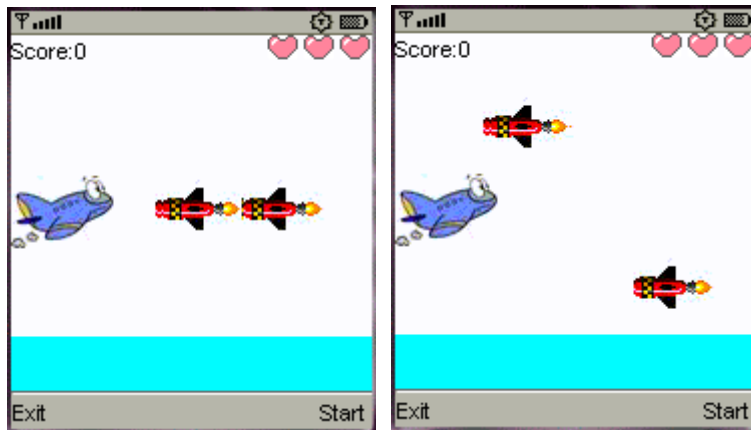
        MissileV[i] = Math.abs(rand.nextInt())%3 + 1;           // Random the speed
    }
    // Draw the missile on screen
    g.drawImage(imgMissile, MissileX[i], MissileY[i], g.TOP|g.LEFT);
}

// Draw the score
g.setColor(0, 0, 0);
g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

// Display the number of life
for (i=1; i<=CurrentLife; i++) {
    g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
}
}
} catch(Exception e) {
    System.out.println(e.getMessage());
}
}
}

```

3. Compile and execute the MIDlet, you can see the missile flying to the airplane.



## 5. Attract the Air plane

1. Open the project “*Shooting*” and modify the source file “*ImageCanvas.java*” in the “*src*” folder (C:\WTK22\apps\Shooting\src).

### ImageCanvas.java

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// import the random library
import java.util.Random;

/*-----*/
/* Canvas for graphic display */
/*-----*/
public class ImageCanvas extends Canvas implements Runnable {
    private Random rand = new Random(); // Random seek
    private Image imgPlane; // Graphic for Plane
    private Image imgLife; // Graphic for Life
    private Image imgOpening; // Graphic for Opening
    private Image imgMissile; // Graphic for Missile
    private int PlaneY = 70; // Y-coordinate for Air plane
    private int i; // Looping variables
    private int CurrentLife = 3; // Current Life
    private int score = 0; // Display the Score
    private int ScreenMode = 0; // Game Screen Status
    private boolean ExitFlag = false; // The Flag to Exit
    private int MissileNumber = 4; // Number of Missile
    private int idx = -1; // Missile that being hit by air plane

    // Missile position (X, Y) and speed V
    private int[] MissileX = new int[10];
    private int[] MissileY = new int[10];
    private int[] MissileV = new int[10];

    // Define the game status
    final int GAME_OPEN = 0, // Opening
            GAME_PLAY = 1, // Playing
            GAME_OVER = 9; // Game Over

    public ImageCanvas() {
        // Initialize the Missile position
        for (i=0; i<MissileNumber; i++) {
            MissileX[i] = 180; // Set the x-coordinate to 180
            MissileY[i] = 0; // Set the y-coordinate to 0
            MissileV[i] = 0; // Set the speed to 0
        }

        // Load the image from resource folder
        try {
            imgPlane = Image.createImage("/plane.png"); // Air Plane
            imgLife = Image.createImage("/life.png"); // Life
            imgOpening = Image.createImage("/opening.png"); // Opening
            imgMissile = Image.createImage("/missile.png"); // Missile
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    // Execute the Thread
    public void run() {
        try {
```

```

// Start the game
ScreenMode = GAME_PLAY;

// Continuous to repaint the screen until the ExitFlag = True
while (!ExitFlag) {
    repaint(); // Call paint() to repaint the screen
    Thread.sleep(100); // Sleep for 0.1 second
}
} catch (Exception e) {
    System.out.println(e.getMessage());
}
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
    // Move the air plane up when press [2]
    if (keyCode == KEY_NUM2) {
        PlaneY = 30;
    // Move the air plane middle when press [5]
    } else if (keyCode == KEY_NUM5) {
        PlaneY = 70;
    // Move the air plane down when press [8]
    } else if (keyCode == KEY_NUM8) {
        PlaneY = 110;
    }
}

// Paint the graphic on the screen
public void paint(Graphics g) {
    try {
        if (ScreenMode == GAME_OPEN) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Display the Opening
            g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
        } else if (ScreenMode == GAME_PLAY) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Draw the ground
            g.setColor(0, 255, 255);
            g.fillRect(0, 150, 180, 180);

            // Draw the Air Plane
            g.drawImage(imgPlane, 0, PlaneY, g.TOP|g.LEFT);

            // Draw the missile
            for (i=0; i<MissileNumber; i++) {
                // Move the missile to attack air plane if speed > 0
                if (MissileV[i] > 0) {
                    if (MissileX[i] > 0) {
                        MissileX[i] = MissileX[i] - MissileV[i];

                        // Handle the event for missile hit the air plane
                        if (MissileX[i] < 50) {
                            // Reduce the Life
                            CurrentLife = CurrentLife - 1;

                            // Reset the Missile position
                            for (i=0; i<MissileNumber; i++) {
                                MissileX[i] = 180; // Set the x-coordinate to 180
                                MissileY[i] = 0; // Set the y-coordinate to 0
                                MissileV[i] = 0; // Set the speed to 0
                            }
                        }

                        // Pause the screen for 2 seconds
                        Thread.sleep(2000);
                    }
                }
            }
        }
    }
}

```

```

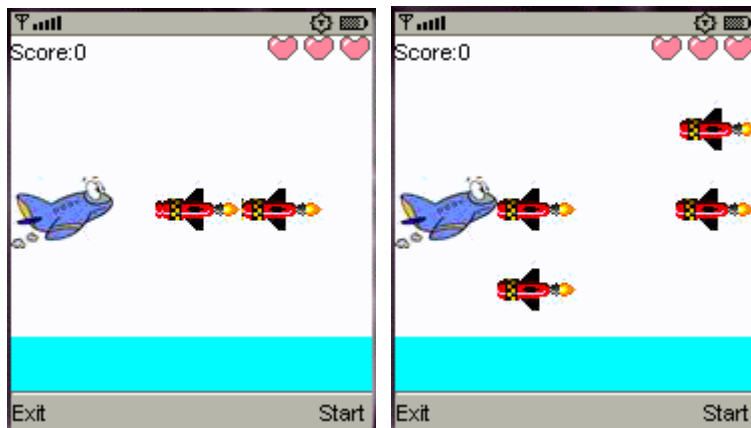
    } else if (PlaneY == MissileY[i] && idx == -1) {
        // Identify the missile being hit
        idx = i;
    }
    } else {
        MissileV[i] = 0;
    }
    } else if (Math.abs(rand.nextInt())%5 == 1) {
        // Create an new missile at 20% chance
        MissileX[i] = 180;
        MissileY[i] = Math.abs(rand.nextInt())%3*40 + 30;    // Random the location
        MissileV[i] = Math.abs(rand.nextInt())%3 + 1;        // Random the speed
    }
    g.drawImage(imgMissile, MissileX[i], MissileY[i], g.TOP|g.LEFT);
}

// Draw the score
g.setColor(0, 0, 0);
g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

// Display the number of life
for (i=1; i<=CurrentLife; i++) {
    g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
}
}
} catch(Exception e) {
    System.out.println(e.getMessage());
}
}
}

```

2. Compile and execute the MIDlet, can you find some bug here?



## 6. Handle the Game Over

1. Open the project “*Shooting*” and put the game over image (PNG format) into the resource folder (C:\WTK22\apps\Shooting\res).



2. Modify the source file “*ImageCanvas.java*” in the “*src*” folder (C:\WTK22\apps\Shooting\src).

### ImageCanvas.java

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// import the random library
import java.util.Random;

/*-----*/
/* Canvas for graphic display */
/*-----*/
public class ImageCanvas extends Canvas implements Runnable {
    private Random rand = new Random(); // Random seek
    private Image imgPlane; // Graphic for Plane
    private Image imgLife; // Graphic for Life
    private Image imgOpening; // Graphic for Opening
    private Image imgMissile; // Graphic for Missile
    private Image imgGameOver; // Graphic for Game over
    private int PlaneY = 70; // Y-coordinate for Air plane
    private int i; // Looping variables
    private int CurrentLife = 3; // Current Life
    private int score = 0; // Display the Score
    private int ScreenMode = 0; // Game Screen Status
    private boolean ExitFlag = false; // The Flag to Exit
    private int MissileNumber = 4; // Number of Missile
    private int idx = -1; // Missile that being hit by air plane

    // Missile position (X, Y) and speed V
    private int[] MissileX = new int[10];
    private int[] MissileY = new int[10];
    private int[] MissileV = new int[10];

    // Define the game status
    final int GAME_OPEN = 0, // Opening
            GAME_PLAY = 1, // Playing
            GAME_OVER = 9; // Game Over

    public ImageCanvas() {
        // Initialize the Missile position
        for (i=0; i<MissileNumber; i++) {
            MissileX[i] = 180; // Set the x-coordinate to 180
            MissileY[i] = 0; // Set the y-coordinate to 0
            MissileV[i] = 0; // Set the speed to 0
        }
    }
}
```

```

// Load the image from resource folder
try {
    imgPlane = Image.createImage("/plane.png");           // Air Plane
    imgLife = Image.createImage("/life.png");           // Life
    imgOpening = Image.createImage("/opening.png");     // Opening
    imgMissile = Image.createImage("/missile.png");     // Missile
    imgGameOver = Image.createImage("/gameover.png"); // Game Over
} catch (Exception e) {
    System.out.println(e.getMessage());
}
}

// Execute the Thread
public void run() {
    try {
        // Start the game
        ScreenMode = GAME_PLAY;

        // Continuous to repaint the screen until the ExitFlag = True
        while (!ExitFlag) {
            repaint(); // Call paint() to repaint the screen
            Thread.sleep(100); // Sleep for 0.1 second
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
    // Move the air plane up when press [2]
    if (keyCode == KEY_NUM2) {
        PlaneY = 30;
    } // Move the air plane middle when press [5]
    else if (keyCode == KEY_NUM5) {
        PlaneY = 70;
    } // Move the air plane down when press [8]
    else if (keyCode == KEY_NUM8) {
        PlaneY = 110;
    }
}

// Paint the graphic on the screen
public void paint(Graphics g) {
    try {
        if (ScreenMode == GAME_OPEN) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Display the Opening
            g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
        } else if (ScreenMode == GAME_PLAY) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Draw the ground
            g.setColor(0, 255, 255);
            g.fillRect(0, 150, 180, 180);

            // Draw the Air Plane
            g.drawImage(imgPlane, 0, PlaneY, g.TOP|g.LEFT);

            // Draw the missile
            for (i=0; i<MissileNumber; i++) {
                // Move the missile to attack air plane if speed > 0
                if (MissileV[i] > 0) {
                    if (MissileX[i] > 0) {

```

```

        MissileX[i] = MissileX[i] - MissileV[i];

        // Handle the event for missile hit the air plane
        if (MissileX[i] < 50) {
            // Reduce the Life
            CurrentLife = CurrentLife - 1;

            // Change the status to Game Over if life = 0
            if (CurrentLife <= 0) {
                ScreenMode = GAME_OVER;
            }

            // Reset the Missile position
            for (i=0; i<MissileNumber; i++) {
                MissileX[i] = 180; // Set the x-coordinate to 180
                MissileY[i] = 0; // Set the y-coordinate to 0
                MissileV[i] = 0; // Set the speed to 0
            }

            // Pause the screen for 2 seconds
            Thread.sleep(2000);
        } else if (PlaneY == MissileY[i] && idx == -1) {
            // Identify the missile being hit
            idx = i;
        }
        } else {
            MissileV[i] = 0;
        }
    } else if (Math.abs(rand.nextInt())%5 == 1) {
        // Create an new missile at 20% chance
        MissileX[i] = 180;
        MissileY[i] = Math.abs(rand.nextInt())%3*40 + 30; // Random the location
        MissileV[i] = Math.abs(rand.nextInt())%3 + 1; // Random the speed
    }
    g.drawImage(imgMissile, MissileX[i], MissileY[i], g.TOP|g.LEFT);
}

// Draw the score
g.setColor(0, 0, 0);
g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

// Display the number of life
for (i=1; i<=CurrentLife; i++) {
    g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
}
} else if (ScreenMode == GAME_OVER) {
    // Display the Game Over and stop the thread
    g.drawImage(imgGameOver, 0, 0, g.TOP|g.LEFT);

    // Set the ExitFlag = true to stop the thread
    ExitFlag = true;
}
} catch (Exception e) {
    System.out.println(e.getMessage());
}
}
}

```

3. Compile and execute the MIDlet, how long can you survive?



## 7. Attract the Missiles by Laser

1. Open the project “*Shooting*” and modify the source file “*ImageCanvas.java*” in the “*src*” folder (C:\WTK22\apps\Shooting\src).

### ImageCanvas.java

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// import the random library
import java.util.Random;

/*-----*/
/* Canvas for graphic display */
/*-----*/

public class ImageCanvas extends Canvas implements Runnable {
    private Random rand = new Random(); // Random seek
    private Image imgPlane; // Graphic for Plane
    private Image imgLife; // Graphic for Life
    private Image imgOpening; // Graphic for Opening
    private Image imgMissile; // Graphic for Missile
    private Image imgGameOver; // Graphic for Game over
    private int PlaneY = 70; // Y-coordinate for Air plane
    private int i; // Looping variables
    private int CurrentLife = 3; // Current Life
    private int score = 0; // Display the Score
    private int ScreenMode = 0; // Game Screen Status
    private boolean ExitFlag = false; // The Flag to Exit
    private int MissileNumber = 4; // Number of Missile
    private int idx = -1; // Missile that being hit by air plane
    private int FireFlag = 0; // Air plane Fire Flag
    private int FireY = 0; // Laser Gun position

    // Missile position (X, Y) and speed V
    private int[] MissileX = new int[10];
    private int[] MissileY = new int[10];
    private int[] MissileV = new int[10];

    // Define the game status
    final int GAME_OPEN = 0, // Opening
            GAME_PLAY = 1, // Playing
            GAME_OVER = 9; // Game Over

    public ImageCanvas() {
        // Initialize the Missile position
        for (i=0; i<MissileNumber; i++) {
            MissileX[i] = 180; // Set the x-coordinate to 180
            MissileY[i] = 0; // Set the y-coordinate to 0
            MissileV[i] = 0; // Set the speed to 0
        }

        // Load the image from resource folder
        try {
            imgPlane = Image.createImage("/plane.png"); // Air Plane
            imgLife = Image.createImage("/life.png"); // Life
            imgOpening = Image.createImage("/opening.png"); // Opening
            imgMissile = Image.createImage("/missile.png"); // Missile
            imgGameOver = Image.createImage("/gameover.png"); // Game Over
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```

// Execute the Thread
public void run() {
    try {
        // Start the game
        ScreenMode = GAME_PLAY;

        // Continuous to repaint the screen until the ExitFlag = True
        while (!ExitFlag) {
            repaint(); // Call paint() to repaint the screen
            Thread.sleep(100); // Sleep for 0.1 second
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
    // Move the air plane up when press [2]
    if (keyCode == KEY_NUM2) {
        PlaneY = 30;
    } // Move the air plane middle when press [5]
    else if (keyCode == KEY_NUM5) {
        PlaneY = 70;
    } // Move the air plane down when press [8]
    else if (keyCode == KEY_NUM8) {
        PlaneY = 110;
    }
    FireFlag = 1;
    FireY = PlaneY;
}

// Paint the graphic on the screen
public void paint(Graphics g) {
    try {
        if (ScreenMode == GAME_OPEN) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Display the Opening
            g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
        } else if (ScreenMode == GAME_PLAY) {
            // Clear the screen
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, 180, 180);

            // Draw the ground
            g.setColor(0, 255, 255);
            g.fillRect(0, 150, 180, 180);

            // Draw the Air Plane
            g.drawImage(imgPlane, 0, PlaneY, g.TOP|g.LEFT);

            // Draw the missile
            for (i=0; i<MissileNumber; i++) {
                // Move the missile to attack air plane if speed > 0
                if (MissileV[i] > 0) {
                    if (MissileX[i] > 0) {
                        MissileX[i] = MissileX[i] - MissileV[i];

                        // Handle the event for missile hit the air plane
                        if (MissileX[i] < 50) {
                            // Reduce the Life
                            CurrentLife = CurrentLife - 1;

                            // Change the status to Game Over if life = 0
                            if (CurrentLife <= 0) {
                                ScreenMode = GAME_OVER;
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        // Reset the Missile position
        for (i=0; i<MissileNumber; i++) {
            MissileX[i] = 180; // Set the x-coordinate to 180
            MissileY[i] = 0;   // Set the y-coordinate to 0
            MissileV[i] = 0;   // Set the speed to 0
        }

        // Pause the screen for 2 seconds
        Thread.sleep(2000);
    } else if (PlaneY == MissileY[i] && idx == -1) {
        // Identify the missile being hit
        idx = i;
    }
    } else {
        MissileV[i] = 0;
    }
    } else if (Math.abs(rand.nextInt())%5 == 1) {
        // Create an new missile at 20% chance
        MissileX[i] = 180;
        MissileY[i] = Math.abs(rand.nextInt())%3*40 + 30; // Random the location
        MissileV[i] = Math.abs(rand.nextInt())%3 + 1;     // Random the speed
    }
    g.drawImage(imgMissile, MissileX[i], MissileY[i], g.TOP|g.LEFT);
}

// Display the Laser
if (FireFlag > 0) {
    FireFlag = FireFlag + 20;
    if (FireFlag < 180) {
        // Draw the Laser Gun Effect
        g.setColor(250, 215, 133);
        g.fillRect(50, FireY, FireFlag, 5);

        // Destroy the missile if the laser hit the missile
        if (idx != -1 && FireFlag+50 > MissileX[idx]) {
            MissileX[idx] = 180; // Reset the location
            MissileV[idx] = 0;   // Reset the speed

            // Increase the source
            score = score + 1;

            // Reset the hit flag
            idx = -1;
        }
    } else {
        FireFlag = 0;
    }
}

// Draw the score
g.setColor(0, 0, 0);
g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

// Display the number of life
for (i=1; i<=CurrentLife; i++) {
    g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
}
} else if (ScreenMode == GAME_OVER) {
    // Display the Game Over and stop the thread
    g.drawImage(imgGameOver, 0, 0, g.TOP|g.LEFT);

    // Set the ExitFlag = true to stop the thread
    ExitFlag = true;
}
} catch(Exception e) {
    System.out.println(e.getMessage());
}
}
}

```

2. Compile and execute the MIDlet, you can use your laser gun to attract the missile, how many mark can you get?

