OBJECT-ORIENTED PROGRAMMING:
DUCK HUNT PROGRAM

For this assignment you are going to create the classic Nintendo video game **Duck Hunt**. In the game Duck Hunt, players use a gun to shoot ducks that are in mid-flight for points. The ducks appear on the screen one or two at a time and the player needs to aim the cursor at the ducks while clicking on the left mouse button to shoot them.

Your version of the game must meet the following requirements:

- When the program begins, a message should appear at the bottom of the screen instructing the user to click the SPACEBAR to begin the game.

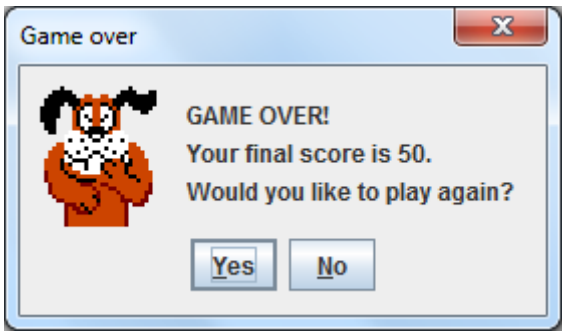
- Once the game begins, there must be 1-2 ducks that fly across the screen at any given time and a dog must walk across the screen. When creating the duck, you must randomly assign the direction it flies (east or west). Also, the ducks must move up and down along the y-axis in addition to left and right without ever going passed the grass. Once a duck goes off the screen, you must randomly assign the direction it's going to fly again and make it appear on the screen.

- Players must shoot as many ducks as possible while maintaining a shooting accuracy above 65%. For every duck the player shoots down, 10 points will be rewarded. When a duck is shot, the image of the duck must change to an image of a duck falling and it must fall into the grass. Once the duck disappears behind the grass, a new duck should be generated.

- The player's accuracy rating goes down whenever they fire a shot and miss a duck and/or when a duck flies across the screen without being hit.

- If the player shoots the dog, the dog must move behind the grass and laugh at the player for 5 seconds. During this time, the player's gun must be disabled preventing the player from shooting any ducks during that interval.

- The game ends once the user's accuracy rating is below 65%. When the game ends, a message dialog should appear indicating that the game is over and prompting the user if they would like to play again. If the user selects YES, the game should restart. If the user selects NO, the program should terminate.




```
int playAgain = JOptionPane.showConfirmDialog(null, "GAME
   OVER!\nYour final score is " + score + ".\n" + "Would you like
   to play again?", "Game over", JOptionPane.YES_NO_OPTION,
   JOptionPane.PLAIN_MESSAGE, new ImageIcon("images\\dog.gif"));
```

```
if (playAgain == JOptionPane.YES_OPTION)
{
	// Restart the game
}
else
{
	// End the game
}
```

For this program you will need to create two classes, a **Dog** class and a **Duck** class, according to the following UML diagram:

Dog
- dogHit: boolean - imgDog, imgLaughingDog: ImageIcon - t: Timer - xPos, yPos, direction, width, height, seconds: int + EAST: int + WEST: int
Dog() Dog(x: int, y: int) Dog(x: int, y: int, dir: int) + draw(g2: Graphics2D): void + move(): void + move(numPixels: int): void + setDirection(dir: int): void + setImage(img: ImageIcon): void + setPosition(x: int, y: int): void + setX(x: int): void + setY(y: int): void + shootDog(): void + getHeight(): int + getImage(): ImageIcon + getWidth(): int + getX(): int + getY(): int + isFacingEast(): boolean + isFacingWest(): boolean + isShot(): boolean + actionPerformed(e: ActionEvent): void

Duck
- dead: boolean - imgDuck, imgFallingDuck: ImageIcon - rnd: Random - xPos, yPos, direction, width, height, seconds: int + EAST: int + WEST: int
Duck() Duck(x: int, y: int, dir: int) Duck(panelWidth: int, panelHeight: int) + draw(g2: Graphics2D): void + moveUp(): void + moveDown(): void + getHeight(): int + getWidth(): int + getX(): int + getY(): int + isFacingEast(): boolean + isFacingWest(): boolean + isDead(): boolean + move(): void + setDirection(dir: int): void + move(xPixels: int, yPixels: int): void + setPosition(x: int, y: int): void + setImage(img: ImageIcon): void + setX(x: int): void + setY(y: int): void + killDuck(): void

Like all the other programs you have created so far, you need to ensure that your program is user-friendly and includes some of the features that users would expect from a video game. Feel free to add any other features to the game. The requirements outlined above are the minimum requirements that your game must fulfill.

Save the project as **Duck Hunt** in your COMPLETED ASSIGNMENTS folder.

OBJECT-ORIENTED PROGRAMMING:
DUCK HUNT PROGRAM

NAME: _____ TOTAL: / 40

CATEGORY	CRITERIA	LEVEL 1 50 – 59%	LEVEL 2 60 – 69%	LEVEL 3 70 – 79%	LEVEL 4 80 – 100%	MARK
Knowledge and Understanding	Demonstrates an understanding of how to create a program using object-oriented programming techniques	• Demonstrates limited understanding of how to create a program using object-oriented programming techniques **5.0-5.9**	• Demonstrates some understanding of how to create a program using object-oriented programming techniques **6.0-6.9**	• Demonstrates considerable understanding of how to create a program using object-oriented programming techniques **7.0-7.9**	• Demonstrates a thorough understanding of how to create a program using object-oriented programming techniques **8.0-10**	**/10**
Thinking	The program meets all the specifications required Validates program to ensure the program produces correct results	• Program meets a limited number of the required specifications • Validates program with limited success **5.0-5.9**	• Program meets some of the required specifications • Validates program with some success **6.0-6.9**	• Program meets most of the required specifications • Validates program with considerable success **7.0-7.9**	• Program meets all of the required specifications • Validates program with great success **8.0-10**	**/10**
Communication	Provides internal documentation that clearly explains the methods and the program logic	• Documents program logic with limited success **5.0-5.9**	• Documents program logic with some success **6.0-6.9**	• Documents program logic with considerable success **7.0-7.9**	• Documents program logic with great success **8.0-10**	**/10**
Application	Effectively applies modular programming concepts and skills to create a Duck Hunt program	• Applies modular programming concepts and skills with limited success **5.0-5.9**	• Applies modular programming concepts and skills with some success **6.0-6.9**	• Applies modular programming concepts and skills with considerable success **7.0-7.9**	• Applies modular programming concepts and skills with great success **8.0-10**	**/10**

CURRICULUM EXPECTATIONS THAT ARE COVERED IN THIS ASSIGNMENT:

A1. Demonstrate the ability to use different data types in computer programs.
A2. Demonstrate the ability to use control structures and simple algorithms in computer programs.
A3. Demonstrate the ability to use subprograms within computer programs.
A4. Use proper code maintenance techniques and conventions when creating computer programs.
B1. Use a variety of problem-solving strategies to solve different types of problems independently and as part of a team.
B2. Design software solutions to meet a variety of challenges.
B3. Design algorithms according to specifications.