Reading 16: Object Orientation

# Exercise 1: Summarize

Object-oriented programming (OOP) involves creating objects with data and methods, featuring classes, prototypes, inheritance, encapsulation, and polymorphism, and can be implemented in various ways across different programming languages.

# Exercise 2: Demonstrate & Explain

These functions make a list of bows and their variables using oop principles. A function is then called to get the average of the bow draw weights. The languages chosen are Java and C.

# Java:

import java.util.ArrayList;

import java.util.List;

public class Bow {

private String riser, limbs, string, style;

private int weight;

private double braceHeight;

private boolean stabilizer, sight, wristStrap;

public Bow(String riser, String limbs, String string, String style, int weight, double braceHeight, boolean stabilizer, boolean sight, boolean wristStrap) {

this.riser = riser;

this.limbs = limbs;

this.string = string;

this.style = style;

this.weight = weight;

this.braceHeight = braceHeight;

this.stabilizer = stabilizer;

this.sight = sight;

this.wristStrap = wristStrap;

}

public String getRiser() { return riser; }

public String getLimbs() { return limbs; }

public String getString() { return string; }

public String getStyle() { return style; }

public int getWeight() { return weight; }

public double getBraceHeight() { return braceHeight; }

public boolean hasStabilizer() { return stabilizer; }

public boolean hasSight() { return sight; }

public boolean hasWristStrap() { return wristStrap; }

public void setRiser(String riser) { this.riser = riser; }

public void setLimbs(String limbs) { this.limbs = limbs; }

public void setString(String string) { this.string = string; }

public void setStyle(String style) { this.style = style; }

public void setWeight(int weight) { this.weight = weight; }

public void setBraceHeight(double braceHeight) { this.braceHeight = braceHeight; }

public void setStabilizer(boolean stabilizer) { this.stabilizer = stabilizer; }

public void setSight(boolean sight) { this.sight = sight; }

public void setWristStrap(boolean wristStrap) { this.wristStrap = wristStrap; }

}

public class Main {

public static void main(String[] args) {

// Array of data to create Bow objects

Object[][] bowData = {

{"Carbon Fiber", "Composite", "Dacron", "Olympic", 45, 7.5, true, true, true},

{"Wood", "Bamboo", "Dacron", "Traditional", 35, 7.0, false, false, false},

{"Aluminum", "Fiberglass", "Horse Hair", "Compound", 50, 6.5, true, true, false}

};

// Create a list of Bow objects

List<Bow> bows = new ArrayList<>();

for (Object[] data : bowData) {

// Create a new Bow object and add it to the list

bows.add(new Bow(

(String) data[0],

(String) data[1],

(String) data[2],

(String) data[3],

(int) data[4],

(double) data[5],

(boolean) data[6],

(boolean) data[7],

(boolean) data[8]

));

}

// Calculate the average weight of the bows

double averageWeight = calculateAverageWeight(bows);

System.out.println("Average weight of bows: " + averageWeight);

}

public static double calculateAverageWeight(List<Bow> bows) {

int totalWeight = 0;

for (Bow bow : bows) {

totalWeight += bow.getWeight();

}

return (double) totalWeight / bows.size();

}

}

# C:

#include <stdio.h>

#include <string.h>

typedef struct {

char riser[50];

char limbs[50];

char string[50];

char style[50];

int weight;

double braceHeight;

int stabilizer;

int sight;

int wristStrap;

} Bow;

Bow createBow(const char\* riser, const char\* limbs, const char\* string, const char\* style, int weight, double braceHeight, int stabilizer, int sight, int wristStrap) {

Bow bow;

strcpy(bow.riser, riser);

strcpy(bow.limbs, limbs);

strcpy(bow.string, string);

strcpy(bow.style, style);

bow.weight = weight;

bow.braceHeight = braceHeight;

bow.stabilizer = stabilizer;

bow.sight = sight;

bow.wristStrap = wristStrap;

return bow;

}

const char\* getRiser(Bow\* bow) { return bow->riser; }

const char\* getLimbs(Bow\* bow) { return bow->limbs; }

const char\* getString(Bow\* bow) { return bow->string; }

const char\* getStyle(Bow\* bow) { return bow->style; }

int getWeight(Bow\* bow) { return bow->weight; }

double getBraceHeight(Bow\* bow) { return bow->braceHeight; }

int hasStabilizer(Bow\* bow) { return bow->stabilizer; }

int hasSight(Bow\* bow) { return bow->sight; }

int hasWristStrap(Bow\* bow) { return bow->wristStrap; }

void setRiser(Bow\* bow, const char\* riser) { strcpy(bow->riser, riser); }

void setLimbs(Bow\* bow, const char\* limbs) { strcpy(bow->limbs, limbs); }

void setString(Bow\* bow, const char\* string) { strcpy(bow->string, string); }

void setStyle(Bow\* bow, const char\* style) { strcpy(bow->style, style); }

void setWeight(Bow\* bow, int weight) { bow->weight = weight; }

void setBraceHeight(Bow\* bow, double braceHeight) { bow->braceHeight = braceHeight; }

void setStabilizer(Bow\* bow, int stabilizer) { bow->stabilizer = stabilizer; }

void setSight(Bow\* bow, int sight) { bow->sight = sight; }

void setWristStrap(Bow\* bow, int wristStrap) { bow->wristStrap = wristStrap; }

double calculateAverageWeight(Bow bows[], int size) {

int totalWeight = 0;

for (int i = 0; i < size; i++) {

totalWeight += getWeight(&bows[i]);

}

return (double) totalWeight / size;

}

int main() {

Bow bows[3];

bows[0] = createBow("Carbon Fiber", "Composite", "Dacron", "Olympic", 45, 7.5, 1, 1, 1);

bows[1] = createBow("Wood", "Bamboo", "Dacron", "Traditional", 35, 7.0, 0, 0, 0);

bows[2] = createBow("Aluminum", "Fiberglass", "Horse Hair", "Compound", 50, 6.5, 1, 1, 0);

double averageWeight = calculateAverageWeight(bows, 3);

printf("Average weight of bows: %.2f\n", averageWeight);

return 0;

}

# Exercise 3: Inquire

How does polymorphism in OOP differ from method overloading?