

Question 1:

```
1 package Question1;
2
3 import java.util.Scanner;
4
5 class Rectangle { 2 usages  ↳ sow-2910
6     private int width; 4 usages
7     private int height; 4 usages
8
9     public Rectangle(int width, int height) { 1 usage  ↳ sow-2910
10        if (width < 0) {
11            System.out.println("Warning: input width is negative!");
12            this.width = 1;
13        } else {
14            this.width = width;
15        }
16
17        if (height < 0) {
18            System.out.println("Warning: input height is negative!");
19            this.height = 1;
20        } else {
21            this.height = height;
22        }
23    }
24
25
26    public void visualize(int width, int height) { 1 usage  ↳ sow-2910
27        for (int i = 0; i < height; i++) {
28            for (int j = 0; j < width; j++) {
29                System.out.print("*");
30            }
31            System.out.println();
32        }
33    }
34}
```

```
5
6     public int getWidth() { 1 usage  ↳ sow-2910
7         return this.width;
8     }
9
10    >   public void setWidth(int width) { this.width = width; }
11
12    >   public int getHeight() { return this.height; }
13
14    >   public void setHeight(int height) { this.height = height; }
15
16 }
17
18
19
20
21
22
23
24
25 D  public class Question1 {  ↳ sow-2910
26 D      public static void main(String[] args) {  ↳ sow-2910
27          System.out.println("Test");
28          Scanner scanner = new Scanner(System.in);
29          for (int i = 0; i < 5; i++) {
30              System.out.println("Enter width and height: ");
31              Rectangle rect = new Rectangle(scanner.nextInt(), scanner.nextInt());
32              rect.visualize(rect.getWidth(), rect.getHeight());
33          }
34          scanner.close();
35      }
36
37 }
```

How this code works:

This program defines a class called Rectangle that models a rectangle with a given width and height. In the constructor, if the user inputs a negative value for width or height, the program prints a warning and automatically sets that value to 1. The visualize() method prints the rectangle using asterisks (*), displaying it according to the given width and height. In the main() method, the program asks the user to enter width and height values five times using a Scanner. For each pair of inputs, a new Rectangle object is created and immediately visualized on the console.

Question 2:

```
package Question2;

import java.util.Scanner;

class Triangle { 2 usages  ↗ sow-2910 *
    private int x;  2 usages
    private int y;  2 usages
    private int z;  2 usages

    public Triangle(int x, int y, int z) { 1 usage  ↗ sow-2910
        this.x = x;
        this.y = y;
        this.z = z;
    }

    >     public int getX() { return this.x; }

    >     public int getY() { return this.y; }

    >     public int getZ() { return this.z; }

    public String verify(int x, int y, int z) { 1 usage  ↗ sow-2910
        if (x + y < z || x + z < y || y + z < x) {
            return "Not triangle";
        } else if (x == y && y == z) {
            return "Equilateral";
        } else if (x == y || x == z || y == z) {
            return "Isosceles";
        } else {
            return "Scalene";
        }
    }
}
```

```
public class Question2 { sow-2910*
    public static void main(String[] args) { sow-2910*
        System.out.println("Test");
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter three sides of the triangle:");
        int x = scanner.nextInt();
        int y = scanner.nextInt();
        int z = scanner.nextInt();
        Triangle triangle = new Triangle(x, y, z);
        System.out.printf("The sides of the triangle is: %d %d %d", triangle.getX(), triangle.getY(), triangle.getZ());
        System.out.println(triangle.verify(x, y, z));
        scanner.close();
    }
}
```

How this code works:

This Java program determines the type of triangle based on its three sides. First, it asks the user to input three integers representing the sides of the triangle. These values are then passed to the Triangle class, which stores them as private attributes x, y, and z. The verify method checks if the given sides can form a valid triangle using the triangle inequality rule. If the sides are valid, it further classifies the triangle as Equilateral, Isosceles, or Scalene depending on the equality of the sides. Finally, the program prints out the sides using the verify() method entered and the corresponding type of triangle.

Question 3:

```
package Question3;

import java.util.Scanner;

class Point { 5 usages □ sow-2910
    private double x; 3 usages
    private double y; 3 usages

    public Point(double x, double y) { 2 usages □ sow-2910
        this.x = x;
        this.y = y;
    }

    @ public double distance(Point target) { 1 usage □ sow-2910
        double delta_x = this.x - target.x;
        double delta_y = this.y - target.y;
        return Math.sqrt(Math.pow(delta_x, 2) + Math.pow(delta_y, 2));
    }
}

▷ public class Question3 { □ sow-2910
▷     public static void main(String[] args) { □ sow-2910
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the first point x and y: ");
        double x1 = scanner.nextDouble();
        double y1 = scanner.nextDouble();
        Point first = new Point(x1, y1);

        System.out.println("Enter the second point x and y: ");
        double x2 = scanner.nextDouble();
```

⚠ 2 ↻

```
        double x2 = scanner.nextDouble();
        double y2 = scanner.nextDouble();
        Point second = new Point(x2, y2);

        double result = first.distance(second);
        System.out.printf("The distance between the first point and the second point is: %.2f",
                           result);
        scanner.close();
    }
}
```

How this code works:

This program defines a class named Point that represents a point in a 2D coordinate system with x and y values. The constructor initializes these coordinates when a new Point object is created. The distance() method calculates the Euclidean distance between the current point and another point using the distance formula. In the main() method, the user is asked to input the coordinates of two points through the console. After both points are created, the program calls the distance() method to compute the distance between them. Finally, it prints the result formatted to two decimal places, showing the distance between the two entered points.

Question 4:

```
package Question4;

import java.util.Scanner;

import java.util.ArrayList;

class Order { 2 usages  ⚡ sow-2910 *
    private final ArrayList<Item> items;  4 usages
    private int ID;  3 usages

    public Order(int ID) { 1 usage  ⚡ sow-2910
        items = new ArrayList<>();
        this.ID = ID;
    }

    public double calculateAverageCost() { 1 usage  ⚡ sow-2910 *
        double sum;
        sum = 0;
        for (Item x : items) {
            sum += x.getPrice();
        }
        return sum / items.size();
    }

    >     public void addItem(Item item) { items.add(item); }

    >     public int getID() { return ID; }

    >     public void setID(int ID) { this.ID = ID; }
```

```
class Item { 5 usages  ↳ sow-2910 *
    private int ID;  3 usages
    private String name;  3 usages
    private double price;  3 usages

    public Item(int ID, String name, double price) { 1 usage  ↳ sow-2910
        this.ID = ID;
        this.name = name;
        this.price = price;
    }

    >   public int getID() { return ID; }

    >   public void setID(int ID) { this.ID = ID; }

    >   public String getName() { return name; }

    >   public void setName(String name) { this.name = name; }

    >   public double getPrice() { return price; }

    >   public void setPrice(double price) { this.price = price; }
}
```

```

77 ▷ public class Question4 { sow-2910
78 ▷     public static void main(String[] args) { sow-2910
79         System.out.println("Test");
80         Scanner scanner = new Scanner(System.in);
81         System.out.print("Enter a new number ID for order: ");
82         int ID = scanner.nextInt();
83         System.out.print("How many items in the order: ");
84         int n = scanner.nextInt();
85         Order order = new Order(ID);
86         for (int i = 1; i < n + 1; i++) {
87             System.out.printf("Please enter the ID for item %d: ", i);
88             int IDitems = scanner.nextInt();
89             scanner.nextLine();
90             System.out.printf("Please enter the name for item %d: ", i);
91             String name = scanner.nextLine();
92             System.out.printf("Please enter the price for item %d: ", i);
93             double price = scanner.nextDouble();
94             scanner.nextLine();
95             Item item = new Item(IDitems, name, price);
96             order.addItem(item);
97         }
98         System.out.printf("You have a new order with ID: %d\n", ID);
99         System.out.printf("In the order, you have %d items\n", n);
100        System.out.printf("The average price in the order is: %.2f", order.calculateAverageCost());
101        scanner.close();
102    }
103 }

```

How this code works:

This program manages an order that contains multiple items. It defines two classes: Order and Item, where Order stores a list of Item objects using an Array List. Each Item has an ID, name, and price, while Order has methods to add items and calculate the average cost. In the main method, the program asks the user to input the order ID and the number of items. Then, it repeatedly takes the information of each item and adds it to the order. Finally, it prints out the order details and the average price of all items entered by the user.

Question 5:

```
package Question5;

import java.util.Objects;
import java.util.Scanner;
import java.util.Random;

class Player { 2 usages sow-2910
    private String choice; 4 usages
    private int bet; 2 usages
    private int wallet; 3 usages

    > public Player(int wallet) { this.wallet = wallet; }

    > public int getBet() { return this.bet; }

    public void setBet() { 1 usage sow-2910
        Scanner scanner = new Scanner(System.in);
        this.bet = scanner.nextInt();
    }

    > public String getChoice() { return this.choice; }

    public void setChoice(int choice) { 1 usage sow-2910
        switch (choice) {
            case 1 -> this.choice = "big";
            case 2 -> this.choice = "small";
            default -> this.choice = "Invalid input";
        }
    }
}
```

```
>     public int getWallet() { return this.wallet; }

>     public void addWallet(int money) { this.wallet += money; }
}

class Dice { 5 usages  ↳ sow-2910
    private int value; 3 usages

>     public Dice() { roll(); }

    public void roll() { 2 usages  ↳ sow-2910
        Random r = new Random();
        this.value = r.nextInt( origin: 1, bound: 6 );
    }

>     public int getValue() { return this.value; }

    public String[] get_img_dice() { 1 usage  ↳ sow-2910
        return switch (this.value) {
            case 1 -> new String[]{"+-----+", " |   ● |   |", "+-----+"};
            case 2 -> new String[]{"+-----+", " | ● |   |", "+-----+"};
            case 3 -> new String[]{"+-----+", " | ● | ● |", "+-----+"};
            case 4 -> new String[]{"+-----+", " | ● | ● | ● |", "+-----+"};
        }
    }
}
```

```
        case 5 -> new String[]{"-----+", "| ● ● |", "| ● |", "-----+", "| ● ● |", "-----+"},  
        "+-----+";  
    case 6 -> new String[]{"-----+", "|| ● ● |", "|| ● ● |", "-----+", "| ● ● |", "-----+"},  
        "+-----+";  
    default -> new String[]{"| Lõi! |"};  
};  
}  
  
class House { 2 usages  ↳ sow-2910  
    private static final byte MIN_SMALL = 4; 1 usage  
    private static final byte MAX_SMALL = 10; 1 usage  
    private static final byte MIN_BIG = 11; 1 usage  
    private static final byte MAX_BIG = 17; 1 usage  
    private final Dice[] dices = new Dice[3]; 12 usages  
    private int wallet; 3 usages  
  
    public House(int wallet) { 1 usage  ↳ sow-2910  
        this.wallet = wallet;  
  
        for (int i = 0; i < dices.length; i++) {  
            dices[i] = new Dice();  
        }  
    }  
  
    public void rollDices() { 1 usage  ↳ sow-2910  
        for (Dice dice : dices) {  
            dice.roll();  
        }  
    }  
}
```

```
        }

    }

    public void printDices() { 1 usage  & sow-2910
        System.out.printf("The dices are: %d %d %d\n",
                           dices[0].getValue(),
                           dices[1].getValue(),
                           dices[2].getValue());
        final int num_lines = 5;
        String[][] Dices_array = new String[3][];

        for (int i = 0; i < dices.length; i++) {
            Dices_array[i] = dices[i].get_img_dice();
        }

        for (int i = 0; i < num_lines; i++) {
            System.out.printf("%s %s %s\n",
                               Dices_array[0][i],
                               Dices_array[1][i],
                               Dices_array[2][i]);
        }
    }

    public int sumDices() { 1 usage  & sow-2910
        int sum = 0;
        for (Dice dice : dices) {
            sum += dice.getValue();
        }
        return sum;
    }
}
```

```
public String checkDicesResult() { 1 usage  ⚡ sow-2910
    int dice1 = dices[0].getValue();
    int dice2 = dices[1].getValue();
    int dice3 = dices[2].getValue();
    int sum = dice1 + dice2 + dice3;
    if (dice1 == dice2 && dice1 == dice3) {
        return "same";
    }
    if (sum >= MIN_SMALL && sum <= MAX_SMALL) {
        return "small";
    }
    if (sum >= MIN_BIG && sum <= MAX_BIG) {
        return "big";
    } else {
        return "";
    }
}

> public int getWallet() { return wallet; }

> public void addWallet(int amount) { this.wallet += amount; }

}
```

```
▶ public class Question5 { sow-2910
▶   public static void main(String[] args) { sow-2910
      Scanner scanner = new Scanner(System.in);
      House house_Wallet_Start = new House( wallet: 1000000);
      System.out.println("Enter the amount of money you have: ");
      int Player_Wallet_Start = scanner.nextInt();
      scanner.nextLine();
      Player player = new Player(Player_Wallet_Start);
      System.out.printf("The house has %d\n", house_Wallet_Start.getWallet());
      System.out.printf("The player has %d\n", Player_Wallet_Start);

      int counter = 1;
      while (player.getWallet() > 0) {
          System.out.printf("Round %d\n", counter);
          System.out.println("How much do you want to bet?");
          player.setBet();
          if (player.getBet() <= player.getWallet()) {

              System.out.printf("You have bet $%d\n", player.getBet());

              System.out.println("What do you want to bet?(1 for big/ 2 for small)");
              int n = scanner.nextInt();
              player.setChoice(n);
              house_Wallet_Start.rollDices();
              house_Wallet_Start.printDices();
              System.out.printf("The sum of 3 dices is %d!\n", house_Wallet_Start.sumDices());
          }
      }
  }
}
```

```
String result_home = house_Wallet_Start.checkDicesResult();
if (result_home.equals("same")) {
    System.out.printf("You lost %d\n", player.getBet());
    player.addWallet(-player.getBet());
    house_Wallet_Start.addWallet(player.getBet());
} else if (!Objects.equals(result_home, player.getChoice())) {
    System.out.printf("You lost %d\n", player.getBet());
    player.addWallet(-player.getBet());
    house_Wallet_Start.addWallet(player.getBet());
} else {
    System.out.printf("You won %d\n", player.getBet());
    player.addWallet(player.getBet());
    house_Wallet_Start.addWallet(-player.getBet());
}

System.out.printf("The house has %d\n", house_Wallet_Start.getWallet());
System.out.printf("The player has %d\n", player.getWallet());
if (player.getWallet() <= 0) {
    System.out.println("You are out of money! Bye!");
    break;
}
scanner.nextLine();
```

```

        do {
            System.out.println("Do you still want to continue to play?(Yes/No)");
            String continue_to_play = scanner.nextLine();
            String answer = continue_to_play.toLowerCase();
            validAnswer = true;
            if (answer.equals("yes")) {
                System.out.println("\n");
                countervalidAnswer = false;
            }
        } while (!validAnswer);
    } else {
        System.out.println("You do not have enough money!");
        System.out.println("The amount of money you bet: " + player.getBet());
        System.out.println("The amount of money you have: " + player.getWallet());
    }
}

scanner.close();
}
}

```

How this code works:

The Player class stores the player's wallet, bet amount, and choice (big or small) and allows the user to set these values through input. The Dice class represents a single dice that can roll a random value from 1 to 6 and display its ASCII image. The House class manages three dices, rolls them, prints their results, and checks whether the total sum is “big,” “small,” or “same.” In the main method, the program lets the user input their money, place bets, and decide whether to continue playing each round.