```java
public class Bar extends Shape {
    public Bar() {
        color = "\033[0;34m";
        lines.add("     ");
        lines.add("+---+");
        lines.add("|BAR|");
        lines.add("+---+");
        lines.add("     ");
    }

    @Override
    public void draw() {
        for (String line : lines) {
            System.out.println(line);
        }
    }

    @Override
    public void drawLine(int index) {
        if (index > lines.size()) {
            System.out.print("   ");
        } else {
            System.out.print(color + lines.get(index) + RESET);
        }
    }
}
```

```java
public class Box implements Lookable, Lockable {
    public boolean locked;

    public Box() {
        locked = true;
    }

    @Override
    public void unlock() {
        System.out.println("You've managed to pick the lock!");
        locked = false;
    }

    @Override
    public void lookAt() {
        System.out.println("You see a plain metal box with a large padlock.");
    }
}

```

```java
import java.util.Scanner;
public class Game {
    private Box box;
    private EscapeRoom escapeRoom;

    public static String getUserInput(String question, String[] possibleAnswers) {
        //Display the question
        System.out.print(question);
        //Create a new scanner to read user input
        Scanner scanner = new Scanner(System.in);
        //Declare a variable to hold user input
        String input;
        // Declare a variable to control while loop
        boolean finished = false;
        do {
            // Reads user input from System.in
            input = scanner.nextLine();

            // Iterates over all possible answers to see if the user's input matches any of them.
            for (String possibleAnswer : possibleAnswers) {
                // Case insensitive string comparison
                if (input.equalsIgnoreCase(possibleAnswer)) {
                    // Sets the loop control variable to 'true' to stop the loop
                    finished = true;
                }
            }
            // If we read the end of this block of code, and the loop is still running, we know the user entered an invalid choice
            if (!finished) System.out.println("Invalid Answer! Try again!");
            // Continue iterating as long as we're not finished
        } while (!finished);
        // Close our scanner, it's no longer needed (frees memory)
        scanner.close();
        // Returns the user's choice converted to lower case
        return input.toLowerCase();
    }
}
```

```java
public class Line extends Shape {
    public Line() {
        color = "\033[0;32m";
        lines.add("  $$  ");
        lines.add("  $$  ");
        lines.add("  $$  ");
        lines.add("  $$  ");
        lines.add("  $$  ");
    }

    @Override
    public void draw() {
        for (String line : lines) {
            System.out.println(line);
        }
    }

    @Override
    public void drawLine(int index) {
        if (index > lines.size()) {
            System.out.print("   ");
        } else {
            System.out.print(color + lines.get(index) + 
RESET);
        }
    }
}
```

```java
import java.util.ArrayList;

public abstract class Shape {

    protected String color;
    public static final String RESET = "\033[0m";
    protected ArrayList<String> lines = new ArrayList<
String>();

    public abstract void draw();

    public abstract void drawLine(int index);

}

```

```java
// Inherited class
public class Square extends Shape {

    private int width;

    public Square(int width) {
        color = "\033[0;35m";
        this.width = width;
        String line = "";
        for (int i = 0; i < width; i++) {
            line += "♥";
        }
        lines.add(line);
        for (int i = 0; i < width - 2; i++) {
            line = "♥";
            for (int j = 0; j < width - 2; j++) {
                line += " ";
            }
            line += "♥";
            lines.add(line);
        }
        line = "";
        for (int i = 0; i < width; i++) {
            line += "♥";
        }
        lines.add(line);
    }

    @Override
    public void draw() {
        for (String line : lines) {
            System.out.println(line);
        }
    }

    @Override
    public void drawLine(int index) {
        if (index > lines.size()) {
            System.out.print("   ");
        } else {
            System.out.print(color + lines.get(index) + RESET);
        }
    }

}
```

```java
public class Diamond extends Shape {
    public Diamond() {
        color = "\033[0;32m";
        lines.add("  ^^  ");
        lines.add(" ^^^^ ");
        lines.add("^^^^^^");
        lines.add(" ^^^^ ");
        lines.add("  ^^  ");
    }

    @Override
    public void draw() {
        for (String line : lines) {
            System.out.println(line);
        }
    }

    @Override
    public void drawLine(int index) {
        if (index > lines.size()) {
            System.out.print("      ");
        } else {
            System.out.print(color + lines.get(index) + RESET);
        }
    }
}
```

```java
public interface Lockable {
    boolean locked = true;
    void unlock();
}

```

```java
public interface Lookable {
    void lookAt();
}

```

```java
import java.util.Scanner;
import java.util.ArrayList;

public class EscapeRoom implements Lockable, Lookable {
    public boolean locked;
    private final Box box = new Box();

    //Make a list of actions the user can do
    ArrayList<String> userActions = new ArrayList<>();

    //The actions the user can choose
    public EscapeRoom() {
        locked = true;
        userActions.add("Look Around");
        userActions.add("Unlock the Box");
        userActions.add("Unlock the Door");
    }

    //The actual game
    public void gamePlay() {
        System.out.println("You wake up in the middle of a locked room.");
        Scanner input = new Scanner(System.in);

        boolean gameOver = false;
        boolean lookAround = false;
        while (!gameOver) {
            System.out.print("What would you like to do? ");
            String command = input.nextLine();

            //Make sure the user chooses a command from the list of commands
            boolean validUserAction = false;
            for (String userAction : userActions) {
                if (userAction.equals(command)) {
                    validUserAction = true;
                    break;
                }
            }

            //If the user puts in an invalid action
            if (!validUserAction) {
                System.out.println("Oops! You can't do that!");
            } else {
                //If the user puts in a valid action
                //Check which action the user would like to do, and execute
                if (userActions.get(1).equals(command) && box.locked) {
                    // This will unlock the box
                    box.unlock();
                    lookAround = false;
                } else if (userActions.get(2).equals(command) && !box.locked) {
                    // This will unlock the door
                    this.unlock();
                    gameOver = true;
                } else if (userActions.get(0).equals(command) && !lookAround) {
                    // Check if the box has already been opened
                    if (box.locked) {
                        box.lookAt();
                    } else {
                        this.lookAt();
                    }
                    lookAround = true;
                } else if (!lookAround) {
                    System.out.println("Oops! You can't do that!");
                }
            }
        }
    }

    @Override
    public void unlock() {
        System.out.println("ENJOY FREEDOM!");
        locked = false;
    }

    @Override
    public void lookAt() {
        System.out.println("Inside the box, you find a key to the door!");
    }
}

```

```java
import java.util.Random;

public class SlotMachine {
    private Shape[] shapes;
    private Shape[] display;
    private int credits;

    private final Random rnd = new Random();

    public SlotMachine() {
        shapes = new Shape[5];
        display = new Shape[5];
    }

    public void addCredits(int amount) {
        credits += amount;
    }
    public boolean hasCredits() {
        return credits > 0;
    }

    //Create a function that selects 5 shapes, displays them and checks if the player wins
    //If the player wins, they get $20. If the player loses, they lose $10.
    public void pullArm() {
        draw();
        boolean win = checkForWin();
        addCredits(win ? 20 : -10);
        if (win)
            System.out.println("YOU WIN 20! :O ($" + credits + " remaining)");
        else
            System.out.println("LOSE $10. BOOHOO :'( ($" + credits + " remaining)");
        displayResults();
    }

    //Check if the player wins
    public boolean checkForWin() {
        return ((shapes[4].getClass() == shapes[1].getClass() && shapes[2].getClass() == shapes[0].getClass()) ||
                (shapes[1].getClass() == shapes[2].getClass() && shapes[3].getClass() == shapes[0].getClass()) ||
                (shapes[3].getClass() == shapes[2].getClass() && shapes[4].getClass() == shapes[1].getClass()));
    }

    public void draw() {
        for (int i = 0; i < 5; i++) {
            int shapeNum = rnd.nextInt(4);
            switch (shapeNum) {
                case 0:
                    shapes[i] = new Bar();
                    break;
                case 1:
                    shapes[i] = new Square(5);
                    break;
                case 2:
                    shapes[i] = new Diamond();
                    break;
                case 3:
                    shapes[i] = new Line();
                    break;
                default:
                    break;
            }
        }
    }

    public void displayResults() {
        // For 5 lines
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print("|");
                shapes[j].drawLine(i);
            }
            System.out.println("|");
        }
    }
}

```

```java
public class Assignment4a {
    public static void main(String[] args) {
        Assignment4a assignment4a = new Assignment4a();
        assignment4a.playGame();
    }

    private void playGame() {
        EscapeRoom escapeRoom = new EscapeRoom();
        escapeRoom.gamePlay();
    }
}

```

```java
public class Assignment4b {
    public static void main(String[] args) {
        Assignment4b assignment4b = new Assignment4b();
        assignment4b.playGame();
    }

    private void playGame() {
        SlotMachine slotMachine = new SlotMachine();
        slotMachine.addCredits(20);
        System.out.println("$20 in credits added! Good luck!");

        // Play next round if has credits
        while (slotMachine.hasCredits()) {
            slotMachine.pullArm();
        }

        System.out.println("Game over! No more credits!");
    }
}

```