Перечисление Java — это тип класса.

Как и в случае с классами, вы можете дать им конструктор, добавить переменные экземпляра и методы и даже реализовать интерфейсы.

Следует иметь в виду, что, в отличие от классов, перечисления не наследуют другие классы и не могут быть расширены (т. е. стать суперклассом).

// Example1.java

public enum Day {

SUNDAY,

MONDAY,

TUESDAY,

WEDNESDAY,

THURSDAY,

FRIDAY,

SATURDAY;

}

// Example1.java

public class DescriptionDay {

Day day;

public DescriptionDay(Day day) { this.day = day; }

public void dayIsNice() {

switch (day) {

case MONDAY:

System.out.println("bad");

break;

case FRIDAY:

System.out.println("better");

break;

case SATURDAY:

case SUNDAY:

System.out.println("best");

break;

default:

System.out.println("so-so");

break;

}

}

public static void main(String[] args) {

String str = "MONDAY";

DescriptionDay description = new DescriptionDay(Day.valueOf(str));

description.dayIsNice();

}

}

---

Каждая константа перечисления всегда неявно является public static final.

Поскольку он статический, мы можем получить к нему доступ, используя имя перечисления.

Поскольку это final, мы не можем создавать дочерние перечисления.

---

Мы можем объявить метод main() внутри перечисления. Следовательно, мы можем вызывать enum непосредственно из командной строки.

// Example2.java

public enum Color {

RED,

GREEN,

BLUE;

public static void main(String[] args) {

Color color = Color.RED;

System.out.println(color);

}

}

---

Все перечисления неявно расширяют класс java.lang.Enum.

Поскольку класс может расширять только одного родителя в Java, перечисление не может расширять ничего другого.

---

Метод toString() переопределен в классе java.lang.Enum, который возвращает имя константы перечисления.

---

enum может реализовать множество интерфейсов.

---

values(), ordinal() и valueOf()

// Example3.java

public enum Color {

RED,

GREEN,

BLUE;

}

// Example3.java

public class Test {

public static void main(String[] args) {

Color arr[] = Color.values();

for (Color col : arr) {

System.out.println(col + " at index " + col.ordinal());

}

System.out.println(Color.valueOf("RED"));

}

}

---

перечисление и методы:

enum может содержать как конкретные методы, так и абстрактные методы.

Если класс перечисления имеет абстрактный метод, то каждый экземпляр класса перечисления должен его реализовать.

// Example4.java

public enum Color {

RED,

GREEN,

BLUE;

private Color() {

System.out.println("Constructor called for : " + toString());

}

public void colorInfo() {

System.out.println("Universal Color");

}

}

// Example4.java

public class Test {

public static void main(String[] args) {

Color color = Color.RED;

System.out.println(color);

color.colorInfo();

}

}

---

Еще пример

// Example5.java

public enum Season {

WINTER("snow"), SPRING("green"), SUMMER("sun"), AUTUMN("rain");

private String value;

private Season(String value){

this.value=value;

}

public String getValue() {

return value;

}

}

// Example5.java

class Example {

public static void main(String args[]){

for (Season s : Season.values()){

System.out.println(s + " "+ s.value);

}

}

}

---

classwork -> если еще есть время

Напишите программу Deal.java, которая должна имитировать раздачу карт для игры в покер.

Программа получает число n, задаваемое с консоли пользователем, и раздает карты на n игроков (по 5 карт каждому) из рассортированной колоды.

Разделяйте пять карт, выданных каждому игроку, пустой строкой.

// Example6.java

public class Deal {

public static void main(String[] args) {

final int cardsForPlayer = 5;

int players;

final Scanner sc = new Scanner(System.in);

final Random random = new Random();

final String[] suits = {"clubs", "diamonds", "hearts", "spades"};

final String[] rank = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace"};

final int numberOfCards = suits.length \* rank.length; // number of cards

while (true) {

System.out.println("Enter the number of players: ");

if (sc.hasNextInt()) {

players = sc.nextInt();

if (cardsForPlayer \* players <= numberOfCards) {

break;

} else {

if (players == 0) {

System.out.println("The game has been terminated.");

break;

} else if (players < 0) {

System.out.println("The number of players cannot be less than 0");

} else {

System.out.println("Too many players!");

}

}

} else {

System.out.println("You have not entered a number, or your number is too large!");

}

}

// deck initialization

String[] deck = new String[numberOfCards];

for (int i = 0; i < rank.length; i++) {

for (int j = 0; j < suits.length; j++) {

deck[suits.length \* i + j] = rank[i] + " " + suits[j];

}

}

// deck shuffling

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

int card = i + (random.nextInt(numberOfCards - i)); // random card in the deck

String temp = deck[card];

deck[card] = deck[i];

deck[i] = temp;

}

// the shuffled deck is displayed

for (int i = 0; i < players \* cardsForPlayer; i++) {

System.out.println(deck[i]);

if (i % cardsForPlayer == cardsForPlayer - 1)

System.out.println();

}

}

}

---

homework

сделать рефактор кода, разнести логику в отдельные классы, enums, методы.

если не успели реализовать это во время практики, то копируем класс в дз на платформу.