Enum Types

Intro

Problem

```
public class CurrencyDenom {
    public static final int PENNY = 1;
    public static final int NICKLE = 5;
    public static final int DIME = 10;
    public static final int QUARTER = 25;
}
```

Problem

- **No Type-Safety**
- **No Meaningful Printing**
- **No namespace**

Solution

- **Enum Types**

Enum Types (Перечисления)

- **Типобезопасные перечисления** (**typesafe enums**) в Java представляют собой классы и являются подклассами абстрактного класса `java.lang.Enum`.
- При этом объекты перечисления инициализируются прямым объявлением без помощи оператора `new`.
- При инициализации хотя бы одного перечисления происходит инициализация всех без исключения оставшихся элементов данного перечисления.
- В операторах `case` используются константы без уточнения типа перечисления, так как его тип определен в `switch`.

Enum definition

```
public enum Day {
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
}
```

Example

Enum

```
public enum Type {
    SCIENCE,
    BELLETRE,
    PHANTASY,
    SCIENCE_FICTION
}
```

Model

```
public class Book {
    private String name;
    private Type bookType;
    private String author;

    public Book(String name, String author, Type type) {
        this.bookType = type;
        this.name = name;
        this.author = author;
    }
}
```

Example

```
public class Program {
    public static void main(String[] args) {
        Book b1 = new Book("War and Peace", "L. Tolstoy", Typ
        System.out.printf("Book '%s' has a type %s", b1.name,

        switch (b1.bookType) {
            case BELLETRE:
                System.out.println("Belletre");
                break;
            case SCIENCE:
                System.out.println("Science");
                break;
            case SCIENCE_FICTION:
                System.out.println("Science fiction");
                break;
            case PHANTASY:
                System.out.println("Phantasy");
                break;
        }
```

Output

```
Book 'War and Peace' has a type Belletre
```

Methods

Methods

- Перечисление как подкласс класса `Enum` может содержать поля, конструкторы и методы, реализовывать интерфейсы.
- Каждый тип enum может использовать методы:
  - `static enumType[] values()`
  - `static T valueOf(Class<T> enumType, String arg)`
  - `static enumType valueOf(String arg)`
  - `int ordinal()`

values(): T[],ordinal(): int

```
public enum Type {
    SCIENCE,
    BELLETRE,
    PHANTASY,
    SCIENCE_FICTION
}
```

values(): T[], ordinal(): int

```
public class Program {
    public static void main(String[] args) {
        Type[] types = Type.values();
        for (Type s : types) {
            System.out.println(s);
        }
        System.out.println(Type.BELLETRE.ordinal());
    }
}
```

Output

```
SCIENCE
BELLETRE
PHANTASY
SCIENCE_FICTION
1
```

Enum item as few fields

Enum

```
public enum Color {
    RED("#FF0000"), BLUE("#0000FF"), GREEN("#00FF00");
    private String code;

    public Color(String code) {
        this.code = code;
    }

    public String getCode() {
        return code;
    }
}
```

Example

```
public class Program {
    public static void main(String[] args) {
        System.out.println(Color.RED.getCode());
        System.out.println(Color.GREEN.getCode());
    }
}
```

```
#FF0000
#00FF00
```

Enum item as Method

Enum

```
public enum Operation {
    SUM {
        public int action(int x, int y) {
            return x + y;
        }
    },
    SUBTRACT {
        public int action(int x, int y) {
            return x - y;
        }
    },
    MULTIPLY {
        public int action(int x, int y) {
            return x * y;
        }
    };

    public abstract int action(int x, int y);
}
```

Example

```
public class Program {
    public static void main(String[] args) {
        Operation op = Operation.SUM;
        System.out.println(op.action(10, 4));
        op = Operation.MULTIPLY;
        System.out.println(op.action(6, 4));
    }
}
```

```
40
24
```