GENERIC OVERVIEW

Product => Component => Motherboard
=> Cpu
=> Ram
=> Gpu
=> Hd
=> Peripheral => Monitor
=> Keyboard
=> Mouse
=> Printer

Transaction => Sale
=> Order

Catalogue => StockCatalogue
=> OrdersCatalogue
=> SalesCatalogue

Cli
mainApp
Utils
CodeGenerator

=> : child (hierarchy)

DETAILED OVERVIEW

```
Product {
        String model
        int releaseYear
        String manufacturer
        double price

        // creates a new product
        public Product(String model, int releaseYear, String manufacturer, double price)
        // returns the preview of the product (model)
        public String preview()
        // returns the price of the product
        public double getPrice()
}

Component extends Product {
        int discount

        // creates a new component
        public Component(String model, int releaseYear, String manufacturer, double price)
        // returns component discount
        public static int getDiscount()
}

Peripheral extends Product {
```

```
        int discount

        // creates a new peripheral
        public Peripheral(String model, int releaseYear, String manufacturer, double price)
        // returns peripheral discount
        public static int getDiscount()
}

Motherboard extends Component {
        enum Socket {
                INTEL,
                AMD
        }

        Socket socket
        int memory
        int sataCount

        // creates a new motherboard
        public Motherboard(String model, int releaseYear, String manufacturer, double price, Socket
socket, int memory, int sataCount)
        // returns string represenation of the class
        public String toString()
}

Cpu extends Component {
        double clock
        int cores
        boolean onboardGraphics

        // creates a new cpu
        public Cpu(String model, int releaseYear, String manufacturer, double price, double clock,
int cores, boolean onboardGraphics)
        // returns string representation of the class
        public String toString()
}

Ram extends Component {
        enum Ddr {
                DDR3,
                DDR4,
                DDR5
        }

        Ddr type
        int size
        int freq

        // creates new ram
        public Ram(String model, int releaseYear, String manufacturer, double price, Ddr type, int
size, int freq)
        // returns string representation of the class
```

```
	public String toString()
}

Gpu extends Component {
	enum ChipsetManufacturer {
		NVIDIA,
		AMD
	}

	ChipsetManufacturer chipsetManufacturer
	int memory

	// creates a new gpu
	public Gpu(String model, int releaseYear, String manufacturer, double price,
ChipsetManufacturer chipsetManufacturer, int memory)
	// returns string representation of the class
	public String toString()
}

Hd extends Component {
	enum StorageType {
		HDD,
		SSD
	}

	StorageType storageType
	double size
	int capacity

	// creates a new hd
	public Hd(String model, int releaseYear, String manufacturer, double price, StorageType
storageType, double size, int capacity)
	// returns string representation of the class
	public String toString()
}

Monitor extends Peripheral {
	enum MonitorType {
		MONITOR,
		TV,
		PORTABLE
	}

	enum Port {
		DP,
		HDMI,
		USBC
	}

	MonitorType monitorType
	int size
	String resolution
```

```
        Port[] ports

        // creates a new monitor
        public Monitor(String model, int releaseYear, String manufacturer, double price,
MonitorType monitorType, int size, String resolution, Port[] ports)
        // returns string representation of the class
        public String toString()
}

Keyboard extends Peripheral {
        enum Connection {
                WIRED,
                WIRELESS
        }

        Connection connection

        // creates a new keyboard
        public Keyboard(String model, int releaseYear, String manufacturer, double price,
Connection connection)
        // returns the string representation of the class
        public String toString()
}

Mouse extends Peripheral {
        Sensor {
                LASER,
                OPTICAL
        }

        Sensor sensor
        Connection connection

        // creates a new mouse
        public Mouse(String model, int releaseYear, String manufacturer, double price, Sensor
sensor, Connection connection)
        // returns the string representation of the class
        public String toString()
}

Printer extends Peripheral {
        enum Colors {
                COLORED,
                BLACKNWHITE
        }

        enum PrinterType {
                LASER,
                INKJET
        }

        PrinterType printerType
```

```
        Colors colors

        // creates a new printer
        public Printer(String model, int releaseYear, String manufacturer, double price, PrinterType
printerType, Colors colors)
        // returns the string representation of the class
        public String toString()
}

Transaction {
        enum TransactionType {
                SALE,
                ORDER
        }

        int code
        Product product
        String name
        String number
        String date
        double price

        // creates a new transaction
        public Transaction(TransactionType transactionType, Product product, String name, String
number, String date, double price)
}

Order extends Transaction {
        enum DeliveryStatus {
                DELIVERED,
                TOBEDELIVERED
        }

        String deliveryDate
        DeliveryStatus deliveryStatus
        String productType
        double originalPrice

        // creates a  new order
        public Order(Product product, String name, String number, String deliveryDate)
        // sets delivery status
        public void setDeliveryStatus(DeliveryStatus deliveryStatus)
        // returns string representation of the class
        public String toString()
}

Sale extends Transaction {
        String productType
        double originalPrice

        // creates a new sale
        public Sale(Product product, String name, String number)
```

```
        // returns the string representation of the class
        public String toString()
}

Catalogue {
        // used to generalize catalogues in utils
}

StockCatalogue implements Catalogue {
        Map<Product, Integer> stock
        Map<Integer, String> categories

        // creates a new stockcatalogue
        public StockCatalogue(Map<Product, Integer> products)
        // return filtered stock based on product type
        public Map<Product, Integer> filter(int filter)
        // adds a product to the catalogue
        public void put(Product product, Integer stock)
}

OrdersCatalogue implements Catalogue {
        List<Order> orderedProducts

        // adds an order to the catalogue
        public void addOrder(Order order)
        // returns the list of the orders
        public List<Order> getOrderCatalogue()
        // removes an order from the catalogue
        public void remove(int code)
        // returns the string representation of the class
        public String toString()
}

SalesCatalogue implements Catalogue {
        List<Sale> soldProducts

        // adds sale to the catalogue
        public void addSale(Sale sale)
        // returns the string representation of the class
        public String toString()
}

mainApp {
        // runs the main program
}

CodeGenerator {
        int productCode
        int orderCode
        int saleCode

        // returns new code for product
```

```
        public static int genCode()
        // returns new code for sale/order
        public static int genCode(TransactionType transactionType)
}

Utils {
        // 0 → prints initial message (for example menu of choices)
        // 1 → prints input prompt
        // 2 → while incorrect input prints errMsg
        // 3 → validates input based on range
        public static String checkInput(String initial, String msg, String errMsg, String range)
        // checks if an array of strings contains a certain string
        public static boolean contains(String[] arr, String tar)
        // returns current date with format: dd-mm-yyyy
        public static String genDate()
        // check If user's input date is valid
        public static String checkDate(String msg, String errMsg)
        // checks whether a string only contains numbers
        public static boolean isNumeric(String string)
        // checks if date is valid
        public static boolean isDateValid(String date)
}

Cli {
        // main logic loop
        public static void loop()
        // overviews products based on user input
        static List<Catalogue> overviewProducts(StockCatalogue stockCatalogue, OrdersCatalogue
ordersCatalogue, SalesCatalogue salesCatalogue)
        // overviews orders based on user input
        static List<Catalogue> overviewOrders(StockCatalogue stockCatalogue, OrdersCatalogue
ordersCatalogue, SalesCatalogue salesCatalogue)
}
```