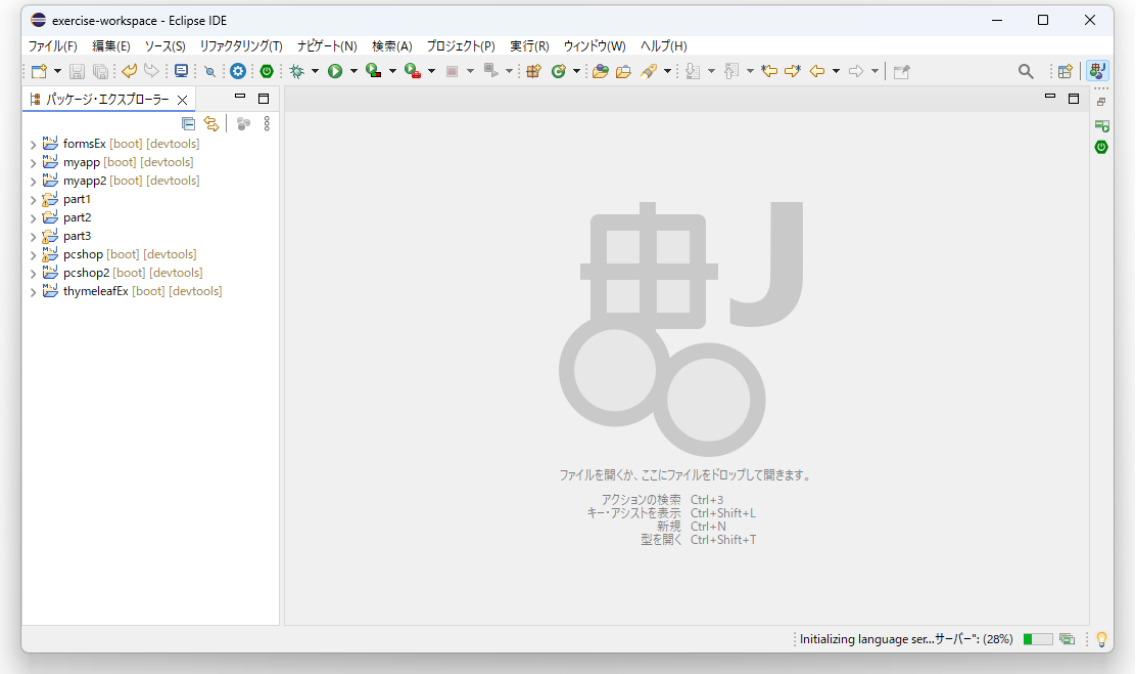
練習問題の解答プロジェクト（exercise-workspace）

exercise-workspace はすべての解答を集めたワークスペースです。
全ての解答プロジェクトが含まれています。

サポートウェブからダウンロードできます。
exercise-workspace-win.zip または exercise-workspace-mac.zip をダウンロードして展開してください。
任意の場所に置いて構いません。ワークスペースに指定して Eclipe を起動します。



内容

Exercise 01　開発ツールの準備と使い方ーWindows

★実行してみるだけの問題なので解答は省略します。

Exercise 02　開発ツールの準備と使い方ーMac OS

★実行してみるだけの問題なので解答は省略します。

Exercise 03　プログラムのかたち

1.①～④に当てはまる語

① パッケージ文

② クラス宣言

③ main

④ メソッド宣言

ブロックの数= 2つ

2.

問1 E

/*/*/* こんにちは */*/*/*

赤字の部分で終端がマッチするので */*/* がエラーになる

問2 1行コメント

//* こんにちは　*//

問3 G

/** から明らか

3.

```
public class Ex3_3 {

    public static void main(String[] args) {
        System.out.print(1000);
        System.out.println("田中宏");
        System.out.print(1010);
        System.out.println("谷川太郎");
        System.out.print(1020);
        System.out.println("鈴木誠");
    }

}
```

Exercise 04　データ型

1.

A. int

B. long

C. byte

D. double

E. char

F. boolean

G. String

2. E

3. B　'¥u0041' が正しい

　G　"ABC" が正しい

　H　"" は空文字として OK だが、''は使えない

Exercise 05　変数宣言と変数の使い方

1.

C　先頭が数字

D　#は使えない

H　-は使えない

I　"は使えない

2.

(1) int a;

(2) double b;

(3) char c;

(4) String d;

(5) boolean e;

3.

```
public class Ex5_3 {
    public static void main(String[] args) {
        double a, b, c, ans;
        a = 1.5;
        b = 8.1;
        c = 2.2;
        ans = a + b + c;
        System.out.println(ans);
    }
}
```

4. C

int より long の方がビット幅が大きいので代入できない

Exercise 06　特殊なリテラル

1.

(1)52　(2)A9　(3)EB　(4)29　(5)C3

2.

```
public class Ex6_2 {

    public static void main(String[] args) {
        System.out.print(1000 + "¥t 田中宏");
        System.out.println(1010 + "¥t 谷川太郎");
        System.out.println(1020+ "¥t 鈴木誠");
    }
}
```

3.

```
public class Ex6_3 {

    public static void main(String[] args) {
        String data = """
                1000田中宏
                1020谷川太郎
                1030鈴木誠
                """;
        System.out.println(data);
    }
}
```

Exercise 07　演算子と演算

1. (2*a + 3*b)*c

2.

A. a+=b;
B. --a;
C. a*=b-1;

3.

A. 10/3
B. 11/3
C. 答えは 102.5
D. 答えは 12.5

4.

(1) D

(2) E　変数宣言では、多重代入はできない

(3) C

(4) E　boolean 型はキャストできない

Exercise 08　いろいろなメソッド

1.

```
package exercise;
import jp.kwebs.lib.Input;
public class Ex8_1 {
    public static void main(String[] args) {
        double a = Input.getDouble("角度");
        double b = Math.toRadians(a);
        System.out.printf("sin = %6.3f%n", Math.sin(b));
        System.out.printf("cos = %6.3f%n" ,Math.cos(b));
    }
}
```

2.

```
import jp.kwebs.lib.Input;
public class Ex8_2 {
    public static void main(String[] args) {
        double a = Input.getDouble("a");
        double b = Input.getDouble("b");
        double c = Input.getDouble("c");
        double d = Math.abs(Math.pow(b, 2) - 4*a*c);
        System.out.printf("ans=%6.2f%n" ,Math.sqrt(d));
    }
}
```

3.

```
public class Ex8_3 {
    public static void main(String[] args) {
        double a = Math.random();
        int p = (int)(6*a+1);
        System.out.println("サイコロの目数＝" + p);
    }
}
```

4.

```
import jp.kwebs.lib.Input;
public class Ex8_4 {
    public static void main(String[] args) {
        int a = Input.getInt("a");
        int b = Input.getInt("b");
        int c = Input.getInt("c");

        int max = Math.max(Math.max(a, b), c);
        System.out.println("a,b,c の中の最大値＝" + max);
    }
}
```

5.

```
import jp.kwebs.lib.Input;
public class Ex8_5 {
    public static void main(String[] args) {
        String str1 = Input.getString("文字列");
        System.out.println("文字数＝" + str1.length());
        System.out.println("先頭から５文字＝" + str1.substring(0,  5));
        System.out.println("先頭から７文字目＝" + str1.charAt(6));

        String str2 = str1.replace("5","0");
        String str3 = str2.toUpperCase();
        System.out.println("str2＝" + str2);
        System.out.println("str3＝" + str3);
    }
}
```

Exercise 09　編集して出力する

1.

```
import jp.kwebs.lib.Input;

public class Ex9_1 {
    public static void main(String[] args) {
        int a = Input.getInt("a");
        double b = Input.getDouble("b");
        String c = Input.getString("c");
        System.out.printf("a=%,8d%nb=%9.3f%nc=%10s%n", a,b,c);
    }
}
```

2.

```
public class Ex9_2 {
   public static void main(String[] args) {

        String json = """
                {
                    "番号":%d
                    "氏名":%s
                    "得点":%d
                }
                """;

        System.out.println(json.formatted(12345, "田中宏", 90));
   }
}
```

テキストブロックでは、エスケープ文字（TAB, "）などもそのまま記述できます。

Exercise 10　データ構造

1.

```
public class Ex10_1 {
   public static void main(String[] args) {

        String[] fruits = {"apple", "banana", "cherry"};

        for(String s : fruits) {
            System.out.println(s);
        }
   }
}
```

2.

```
public class Ex10_2 {

   public static void main(String[] args) {
        double[] data = {0.3, 1.05, 2.2};
        double total = 0;

        for(double x : data) {
            total += x;
        }
        System.out.printf("合計=%.2f%n", total);
        System.out.printf("平均=%.2f%n" ,total/data.length);
   }

}
```

3.

```
import java.util.List;
public class Ex10_3 {

   public static void main(String[] args) {
        var data = List.of(15.1, 8.75, 10.2);
        double total = 0;

        for(double x : data) {
            total += x;
        }
        System.out.println("合計=" + total);
        System.out.println(data);
   }
}
```

4.

```
import java.util.ArrayList;
public class Ex10_4 {

    public static void main(String[] args) {

        var fruits = new ArrayList<String>();
        fruits.add("リンゴ");
        fruits.add("バナナ");
        fruits.add("みかん");

        for(String s : fruits) {
            System.out.println(s);
        }
    }
}
```

5.

```
import java.util.ArrayList;

public class Ex10_5 {

    public static void main(String[] args) {

        var list = new ArrayList<String>();
        list.add("rabbit");
        list.add("cat");
        list.add("dog");

        list.add("bear");// ①
        System.out.println(list.get(1));           // ②
        System.out.println(list.contains("cat"));  // ③
        System.out.println(list.remove(0));        // ④
        System.out.println(list.size());           // ⑤
    }
}
```

6.

```
public record Product(String code, String name, int price, boolean stock) {}
```

7.

```
public class Ex10_7 {
    public static void main(String[] args) {

        Product p = new Product("MT890", "ステンレスネジ", 280, false);

         System.out.println("商品コード = " + p.code());
         System.out.println("商品名     = " + p.name());
         System.out.println("価格       = " + p.price());
         System.out.println("在庫       = " + p.stock());
    }
}
```

8.

```
import java.util.List;

public class Ex10_8 {

    public static void main(String[] args) {
        var items = List.of(
                new Product("MT890", "ステンレスネジ", 280, false),
                new Product("MT810", "タッピングネジ", 160, true),
                new Product("MT900", "スクリューネジ", 330, true));

        for(var p : items) {
            System.out.println(p.name() + "¥t" + p.price());
        }
    }

}
```

9.

```
import java.util.List;
public class Ex10_9 {

    public static void main(String[] args) {
         var cities = List.of(
                 new Population("北海道", 5250, -6.8),
                 new Population("東京都", 13921, 7.1),
                 new Population("大阪府", 8809, -0.4),
                 new Population("福岡県", 5104, -0.7),
                 new Population("沖縄県", 1453, 3.9) );

         for(Population p : cities) {
             System.out.println(p.prefecture() + "(" + p.rate() + ")");
         }
    }
}
```

Exercise 11 メソッドの作成

1. D

戻り値型、引数の数、型、並び順が同じかどうかで判断する

2.

(1)

```
import jp.kwebs.lib.Input;
public class ExII_2_1 {
   public static void main(String[] args) {
        double mile = Input.getDouble("マイル");
        mileToKilo(mile);
   }
   public static void mileToKilo(double mile) {
        System.out.println(mile + "マイルは" + mile * 1.609 + "キロメートル");

   }
}
```

(2)

```
import jp.kwebs.lib.Input;
public class ExII_2_2 {
   public static void main(String[] args) {

        double weight = Input.getDouble("体重(kg)");
        double height = Input.getDouble("身長(cm)");
        bmi(weight, height);
   }

   public static void bmi(double weight, double height) {
        double bmiIndex = weight / Math.pow(height/100, 2);
        System.out.println(bmiIndex);
   }
}
```

(3)

```
import jp.kwebs.lib.Input;
public class ExII_2_3 {
    public static void main(String[] args) {

        double weight = Input.getDouble("体重(kg)");
        double height = Input.getDouble("身長(cm)");
        double bmiIndex = bmi(weight, height);
        System.out.println(bmiIndex);
    }

    public static double bmi(double weight, double height) {
        return weight / Math.pow(height/100, 2);
    }
}
```

(4)

```
import jp.kwebs.lib.Input;
public class ExII_2_4 {
    public static void main(String[] args) {
        String name = Input.getString("氏名");
        int score = Input.getInt("得点");
        String msg = message(name, score);
        System.out.println(msg);
    }
    public static String message(String name, int score) {
        String message = """
                こんにちは%s さん。
                あなたの得点は%d 点でした。
                """;
        return message.formatted(name, score);
    }
}
```

3. D

4. 解答を省略

Exercise 12　いろいろな引数と戻り値

1.

```
import java.util.List;
public class Ex12_1 {
    public static void main(String[] args) {

        var scores = List.of(15, 22, 30, 8, 13);
        int total = sum(scores);
        System.out.println("合計=" + total);
    }

    public static int sum(List<Integer> scores) {
        int total=0;
        for(int score : scores) {
            total += score;
        }
        return total;
    }
}
```

(2)

```
import jp.kwebs.lib.Input;
public class Ex12_2 {
    public static void main(String[] args) {
        var product = getProduct();
        System.out.println(product);
    }
    public static Product getProduct() {
        String code = Input.getString("商品コード");
        String name = Input.getString("商品名");
        int price = Input.getInt("価格");
        boolean missing = Input.getBoolean("在庫");

        return new Product(code, name, price, missing);
    }
}
```

(3)

```
import java.util.List;
public class Ex12_3 {
    public static void main(String[] args) {
        var list = getProductList();
        for(Product p : list) {
            System.out.println(p);
        }
    }
    public static List<Product> getProductList(){

        var list = List.of(
                new Product("MT890", "ステンレスネジ", 280, false),
                new Product("MT810", "タッピングネジ", 160, true),
                new Product("MT900", "スクリューネジ", 330, true));
        return list;
    }
}
```

Exercise 13　条件の作成

1.

(1) n>=100

(2) n>=100 && n<500

(3) n%2==0

(4) (n%3==0 || n%2!=0) && n<100

※ n%2!=0 は n%2==1　でもよい

(5) c>'t'+1

(6) Math.sqrt(x)>2.0

(7) !s.equals("abc");

2.

(1) ①

(2) ③

(3) ①

(4) ①

(5) ③

(6) ②

(7) ②

3.

```
import jp.kwebs.lib.Input;

public class Ex13_3 {
   public static void main(String[] args) {
       String s = Input.getString();
       String disp = s.equals(”OK”) || s.equals(”ok”) ? ”おめでとう” : ”残念” ;
       System.out.println(disp);
   }
}
```

Exercise 14　if 文

1.

```
import jp.kwebs.lib.Input;
public class Ex14_1 {
    public static void main(String[] args) {
        int nin = Input.getInt("人数");
        int gaku = nin * 1000;
        if(nin>=5) {
            gaku *= 0.7;
        }
        System.out.printf("%,d 円", gaku);
    }
}
```

2.

```
import jp.kwebs.lib.Input;
public class Ex14_2 {
    public static void main(String[] args) {
        int y = Input.getInt("西暦年");

        if (y % 4 == 0 && y % 100 != 0 || y % 400 == 0) {
            System.out.println("うるう年です");
        } else {
            System.out.println("うるう年ではありません");
        }
    }
}
```

3.

```
import jp.kwebs.lib.Input;
public class Ex14_3 {
    public static void main(String[] args) {
        int m = Input.getInt("月");

        if(m<1 || m>12) {
            System.out.println("月の値が不正です");
        }else if(m==12 || m<3) {
            System.out.println("冬");
        }else if(m<6) {
            System.out.println("春");
        }else if(m<9) {
            System.out.println("夏");
        }else {
            System.out.println("秋");
        }
    }
}
```

4.

```
public class Ex14_4 {
    public static void main(String[] args) {
        int a = 0, b = 0, c = 0, d = 0;
        double[] height = { 175.2, 160.0, 153.6, 177.5, 185.7, 172.3, 191.3 };
        for(double h : height) {
            if(h < 160) {
                ++a;
            } else if(h < 170) {
                ++b;
            } else if(h < 180) {
                ++c;
            } else {
                ++d;
            }
        }
        System.out.println("A: " + a);
        System.out.println("B: " + b);
        System.out.println("C: " + c);
        System.out.println("D: " + d);
    }
}
```

Exercise 15　switch 文と switch 式

1.

```
import jp.kwebs.lib.Input;
public class Ex15_1 {
    public static void main(String[] args) {
        int m = Input.getInt("月");
        switch(m) {
            case    12,1,2  ->  System.out.println("冬");
            case    3,4,5   ->  System.out.println("春");
            case    6,7,8   ->  System.out.println("夏");
            default ->  System.out.println("秋");
        }
    }
}
```

2.

```
import jp.kwebs.lib.Input;
public class Ex15_2 {
    public static void main(String[] args) {
        int m = Input.getInt("月");
        String season = switch(m) {
            case    12, 1, 2    -> "冬";
            case    3, 4, 5     -> "春";
            case    6, 7, 8     -> "夏";
            default             -> "秋";
        };
        System.out.println(season);
    }
}
```

3.

```
import jp.kwebs.lib.Input;

public class Ex15_3 {
    public static void main(String[] args) {
        String code = Input.getString("商品コード");

        int tanka = switch(code) {
            case    "a100", "b100", "d100"  ->  100;
            case    "a110", "c100"          ->  200;
            case    "b110", "C110"          ->  210;
            case    "b120"                  ->  250;
            default                         ->  0;
        };

        if(tanka!=0) {
            int kosuu = Input.getInt("個数");
            System.out.println("合計金額=" + tanka * kosuu);
        }else {
            System.out.println("商品コードが間違っています");
        }
    }
}
```

Exercise 16　繰り返し構文

1.

```
import jp.kwebs.lib.Input;

public class Ex16_1 {
    public static void main(String[] args) {

        int total = 0;
        for(int i=0; i<5; i++) {
            int number = Input.getInt("整数");
            total += number;
        }
        System.out.println("合計=" + total);
    }
}
```

2.

```
import jp.kwebs.lib.Input;
public class Ex16_2 {
    public static void main(String[] args) {
        String buffer = "";
        while(true) {
            String s = Input.getString();
            if(s==null) {
                break;
            }
            buffer += s;
        }
        System.out.println(buffer);
    }
}
```

3.

```
import jp.kwebs.lib.Input;
public class Ex16_3 {
    public static void main(String[] args) {
        int total=0, number;
        while((number=Input.getInt())!=0) {
            total += number;
        }
        System.out.println("合計" + total);
    }
}
```

4.

```
import java.util.List;

public class Ex16_4 {
    public static void main(String[] args) {
        var names = List.of("apple", "Blackberry", "Lime",  "Mango", "Watermelon");

        for(String str : names) {
            if(str.length()<6) {
                continue;
            }
            System.out.println(str);
        }
   }
}
```

5.

```
import java.util.List;

// 同じファイル内に定義するには public を付けない
record Score (String name, int score){}

public class Ex16_5 {
   public static void main(String[] args) {
        var scores = List.of(
                new Score("田中", 85),
                new Score("鈴木", 66),
                new Score("斎藤", 82),
                new Score("木村", 57),
                new Score("山下", 77) );

        for(Score s : scores) {
            if(s.score()<60) {
                System.out.println("60 点未満です");
                break;
            }
            System.out.println(s);
        }
     }
}
```

Exercise 17　＜問題はありません＞

Exercise 18　オブジェクトの作成

1.

```
import java.util.Objects;

public class BankAccount {
    private String number;          // 口座番号
    private String name;            // 口座名義
    private double balance;         // 残高

    public BankAccount(String number, String name, double balance) {
        this.number = number;
        this.name = name;
        this.balance = balance;
    }
    public BankAccount() {}

    public void deposit(double amount) {     // 2 (1)
        this.balance += amount;
    }
    public void withdraw(double amount) {    // 2 (2)
        this.balance -= amount;
    }

    public String getNumber() {
        return number;
    }
    public String getName() {
        return name;
    }
    public double getBalance() {
        return balance;
    }
    public void setNumber(String number) {
        this.number = number;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setBalance(double balance) {
        this.balance = balance;
    }
    @Override
    public String toString() {
        return "BankAccount [number=" + number + ", name=" + name + ", balance=" + balance + "]";
    }
    @Override
    public int hashCode() {
        return Objects.hash(balance, name, number);
    }
```

```
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        BankAccount other = (BankAccount) obj;
        return Double.doubleToLongBits(balance) == Double.doubleToLongBits(other.balance)
                && Objects.equals(name, other.name) && Objects.equals(number, other.number);
    }
}
```

3.

```
public class Ex18_1 {

    public static void main(String[] args) {

        var account = new BankAccount("12345678", "田中宏", 1000);
        System.out.println(account);
        account.deposit(500);
        account.withdraw(250);
        System.out.println("残高: " + account.getBalance());
    }
}
```

Exercise 19　コンストラクタとクラスメンバ

1.

```
public class Function {
    private int a;
    private int b;

    public Function(int a, int b) {
        this.a = a;
        this.b = b;
    }
    public Function(int b) {
        this(10, b);
    }
    public Function() {
        this(10, 5);
    }
    public void result() {
        System.out.println(a + "x + " + b);
    }

    public static void main(String[] args) {      // 2.
        var f = new Function(3);
        f.result();
    }
}
```

Exercise 20　参照と不変性

1. No

　理由：　フィールド変数の値を増減しなくてはいけないから。

2.

```
public final class Stock {
    private final Product product;    // 商品
    private final int number;         // 在庫数

    public Stock(Product product, int number) {
        this.product = new Product(product.getNumber(), product.getPrice(), product.getName());
        this.number  = number;
    }
    public Product getProduct() {
        return new Product(product.getNumber(), product.getPrice(), product.getName());
    }
    public int getNumber() {
        return number;
    }
```

Exercise 21　クラス継承

1.

```
public class Employee {
 private String name;       // 従業員名
 private String department; // 部署
 private double salary;     // 給与

public Employee(String name, String department, double salary) {
     this.name = name;
     this.department = department;
     this.salary = salary;
 }
public void work() {
     System.out.println(name + ”さんは” + department + ”に所属しています”);
 }

public String getName() {
     return name;
 }
public String getDepartment() {
     return department;
 }
public double getSalary() {
     return salary;
 }
}
```

```
public class Manager extends Employee {
   private int teamSize; // チームの人数（追加フィールド）

   public Manager(String name, String department, double salary, int teamSize) {
       super(name, department, salary);
       this.teamSize = teamSize;
   }

   public int getTeamSize() {
       return teamSize;
   }
}
```

```
public class Developer extends Employee {
   private String mainLanguage; // 主な開発言語（追加フィールド）

   public Developer(String name, String department, double salary, String mainLanguage) {
       super(name, department, salary);
       this.mainLanguage = mainLanguage;
   }
   public String getMainLanguage() {
       return mainLanguage;
   }
}
```

2.

```
public class Ex2I_2 {
    public static void main(String[] args) {
        var manager = new Manager(”田中宏”, ”営業”, 500000, 6);
        System.out.println(manager.getName());
        System.out.println(manager.getDepartment());
        System.out.println(manager.getSalary());
        System.out.println(manager.getTeamSize());
        manager.work();
    }
}
```

3. （B）

Foo クラスのメンバを直接アクセスできない。

Exercise 22　スーパークラスの使い方

1.

```
public final class Stock {
    private final Product product;        // 商品
    private final int number;         // 在庫数

    public Stock(Product product, int number) {
        this.product = new Product(product.getProductNumber(), product.getPrice(), product.getName());
        this.number  = number;
    }
    public Product getProduct() {
        return new Product(product.getProductNumber(), product.getPrice(), product.getName());
    }
    public int getNumber() {
        return number;
    }
    // 生成した移譲メソッド
    public int getProductNumber() {
        return product.getProductNumber();
    }
    public int getPrice() {
        return product.getPrice();
    }
    public String getName() {
        return product.getName();
    }
}
```

2.

```
public class Manager extends Employee {
    private int teamSize; // チームの人数（追加フィールド）

    public Manager(String name, String department, double salary, int teamSize) {
        super(name, department, salary);
        this.teamSize = teamSize;
    }

    @Override
    public void work() {
        System.out.println(getName() + ”さんの部署は”
                    + getDepartment() + ”で、” + teamSize + ”人のチームです”);
    }
    public int getTeamSize() {
        return teamSize;
    }
}
```

```
public class Developer extends Employee {
    private String mainLanguage; // 主な開発言語（追加フィールド）

    public Developer(String name, String department, double salary, String mainLanguage) {
        super(name, department, salary);
        this.mainLanguage = mainLanguage;
    }

    @Override
    public void work() {
        System.out.println(getName() + "さんの部署は"
                + getDepartment() + "で、" + mainLanguage + "で開発します");
    }
    public String getMainLanguage() {
        return mainLanguage;
    }
}
```

3.

```
import java.util.List;
public class Ex22_3 {

    public static void main(String[] args) {
        var list = List.of(
                new Manager("田中宏", "営業", 500000, 6),
                new Developer("藤田一郎", "開発", 500000, "Java"),
                new Developer("斎藤裕子", "開発", 500000, "Python"));

        for(var m : list) {
            m.work();
        }
    }
}
```

Exercise 23　抽象クラスと多態性

1.

```
public abstract class Employee {
 private String name;       // 従業員名
 private String department; // 部署
 private double salary; // 給与

 // コンストラクタ
 public Employee(String name, String department, double salary) {
     this.name = name;
     this.department = department;
     this.salary = salary;
 }

 // 業務を行うメソッド（サブクラスでオーバーライド）
 public abstract void work();

 // ゲッター
 public String getName() {
     return name;
 }
public String getDepartment() {
     return department;
 }
public double getSalary() {
     return salary;
 }
}
```

2.　＜解答省略＞

3.

```
class Root {}
class Foo extends Root {}
class Bar extends Root {}
class Baz extends Root {}
public class Ex23_3 {
   public static void main(String[] args) {
       Root r = new Bar();
       switch(r) {
           case Foo foo -> System.out.println("Foo 型です");
           case Bar bar -> System.out.println("Bar 型です");
           case Baz baz -> System.out.println("Baz 型です");
           default      -> System.out.println("Root の派生型ではありません");
       }
   }
}
```

Exercise 24　インタフェース

1.

```
    public interface Greeter {
        String greet();
    }
```

```
public class Robot implements Greeter {
    @Override
    public String greet() {
        return "こんにちは";
    }
}
```

```
public class Car implements Greeter {
    @Override
    public String greet() {
        return "Hello";
    }
}
```

```
public class Computer implements Greeter {
    @Override
    public String greet() {
        return "Guten Tag";
    }
}
```

```
import java.util.List;
public class Ex24_1 {

    public static void main(String[] args) {
        var fellow = List.of(
                new Robot(),
                new Car(),
                new Computer());

        fellow.forEach(m->System.out.println(m.greet()));
    }
}
```

2.

```
    interface SimpleHasher {
        int computeHash(String input);
    }
```

```
public class SumHasher implements SimpleHasher {

    @Override
    public int computeHash(String input) {
         int total = 0;
         for (char c : input.toCharArray()) {
             total += c;
         }
         return total;
    }
}
```

```
public class XorHasher implements SimpleHasher {

    @Override
    public int computeHash(String input) {
         int result = 0;
         for (char c : input.toCharArray()) {
             result ^= c;
         }
         return result;
    }
}
```

```
import jp.kwebs.lib.Input;
public class Ex24_2 {
    public static void main(String[] args) {
        String input = Input.getString();
    //  SimpleHasher sh = new SumHasher();
        SimpleHasher sh = new XorHasher();

        int hashCode = calculateHash(input, sh);
        System.out.println(input + ": " + hashCode);
    }
    public static int calculateHash(String input, SimpleHasher sh) {
        int hashCode = sh.computeHash(input);
        return hashCode;
    }
}
```

Exercise 25　ラムダ式

1. ウ

   return は不要

2.

(1) B

(2) E

(3) A

(4) D

Exercise 26　コレクションフレームワーク

1.

```
import java.util.List;
public class Ex26_1 {

    public static void main(String[] args) {
        var numbers = List.of(10, 15, 20, 7, 18);
        numbers.forEach(n->System.out.print(n + " "));
    }
}
```

2.

```
import java.util.HashMap;
public class Ex26_2 {
    public static void main(String[] args) {
        var map = new HashMap<Integer, String>();
        map.put(1, "Tokyo");
        map.put(2, "Osaka");
        map.put(3, "Fukuoka");
        map.forEach((k, v)-> System.out.println(k + ": " + v));
    }
}
```

3. B

ハッシュセットは同じ要素はひとつしか持てない

4.

A. LinkedHashSet

B. TreeSet

C. LinkedHashMap

D. TreeMap

5.

```
import java.util.ArrayList;
import java.util.Comparator;

public class Ex26_5 {
    record PhoneNumber (String number, String owner) {}

    public static void main(String[] args) {
        var numbers = new ArrayList<PhoneNumber>();

        numbers.add(new PhoneNumber("12-123-4567", "Tanaka"));
        numbers.add(new PhoneNumber("34-567-8901", "Suzuki"));
        numbers.add(new PhoneNumber("45-678-9012", "Satho"));
        numbers.add(new PhoneNumber("56-789-0123", "Inoue"));
        numbers.add(new PhoneNumber("78-901-2345", "Nakamura"));

        numbers.sort(Comparator.comparing(PhoneNumber::number));
        numbers.forEach(System.out::println);

        System.out.println("データ件数=" + numbers.size());

        numbers.sort(Comparator.comparing(PhoneNumber::owner).reversed());
        numbers.forEach(System.out::println);

    }
}
```

Exercise 27　ストリーム処理

1.

```
public class Ex27_1 {
    public static void main(String[] args) {
        var breads = Bread.getBreadList();
        breads.stream()
              .filter(b->b.country().equals("日本"))
              .forEach(System.out::println);
    }
}
```

2.

```
public class Ex27_2 {
    public static void main(String[] args) {
        var breads = Bread.getBreadList();
        breads.stream()
              .filter(b->b.soldout())
              .map(Bread::name)
              .forEach(System.out::println);
    }
}
```

3.

```
import java.util.Comparator;
public class Ex27_3 {
    public static void main(String[] args) {
        var breads = Bread.getBreadList();
        breads.stream()
              .filter(b->b.country().equals("フランス"))
              .sorted(Comparator.comparing(Bread::price))
              .map(b->b.name()+" "+b.price())
              .forEach(System.out::println);
    }
}
```

4.

```
import java.util.Comparator;
public class Ex27_4 {
    public static void main(String[] args) {
        var breads = Bread.getBreadList();
        breads.stream()
              .sorted(Comparator.comparing(Bread::price).reversed())
              .limit(3)
              .map(b->b.name()+" "+b.price())
              .forEach(System.out::println);
    }
}
```

5.

```
public class Ex27_5 {
    public static void main(String[] args) {
        var breads = Bread.getBreadList();
        breads.stream()
              .map(Bread::country)
              .distinct()
              .sorted()
              .forEach(System.out::println);
    }
}
```

6.

```
public class Ex27_6 {
    public static void main(String[] args) {
        var breads = Bread.getBreadList();
        var average = breads.stream()
              .mapToInt(Bread::price)
              .average();
        System.out.printf("平均＝%.1f%n" ,average.getAsDouble());
    }
}
```

Exercise 28　collectによる終端操作

1.

```
public class Ex28_1 {
    public static void main(String[] args) {

        var data = Sales.getList();
        int sum = data.stream()
            .mapToInt(s->s.quantity() *s.pc().price())
            .sum();
        System.out.printf("総売り上げ = ¥¥%,d%n",sum);
    }
}
```

2.東京支社の総売り上げ高はいくらですか

```
public class Ex28_2 {
    public static void main(String[] args) {

        var data = Sales.getList();
        int sum = data.stream()
            .filter(s->s.office().equals("東京"))
            .mapToInt(s->s.quantity() *s.pc().price())
            .sum();
        System.out.printf("総売り上げ = ¥¥%,d%n",sum);
    }
}
```

3.大阪支社で売り上げたパソコンの名前を重複を除いて列挙してください

```
import java.util.stream.Collectors;
public class Ex28_3 {
    public static void main(String[] args) {

        var data = Sales.getList();
        String names = data.stream()
            .filter(s->s.office().equals("大阪"))
            .map(s->s.pc().name())
            .distinct()
            .collect(Collectors.joining(","));

        System.out.println("大阪支社の機種リスト = " + names);
    }
}
```

4.100,000 円以上の取引は何件ありますか

```
public class Ex28_4 {
    public static void main(String[] args) {

        var data = Sales.getList();
        long count = data.stream()
            .filter(s->s.quantity() * s.pc().price() >=100000)
            .count();

        System.out.printf("100,000 円以上の受注件数 = %d 件%n",count);
    }
}
```

5.担当者別の売上金額の合計を表示してください

```
import java.util.stream.Collectors;
public class Ex28_5 {
    public static void main(String[] args) {

        var data = Sales.getList();
        var map = data.stream()
            .collect(Collectors.groupingBy(Sales::name,
                    Collectors.summingInt(s->s.quantity() * s.pc().price())));

        System.out.println("担当者別売上高");
        map.forEach((k,v) -> System.out.printf(" %s: ¥¥%,d%n" ,k,v));
    }
}
```

Exercise 29　例外処理

1. C

　実行時例外（非チェック例外）なら throws 宣言は不要

2.次のプログラムは青字のメソッドがチェック例外を投げるので、コンパイルエラーになっています。例外処理を追加して正しく動作するようにしてください。

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.List;

public class Ex29_2 {
   public static void main(String[] args) {
        var data = read();
        data.forEach(System.out::println);
   }
   public static List<String> read() {
        Path p = Path.of("someText.txt");
        List<String> lines = null;
        try {
            lines = Files.readAllLines(p);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return lines;
   }
}
```

3.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
public class Ex29_3 {
    public static void main(String[] args) {
        String line = read();
        System.out.println(line);
    }
    public static String read() {
        Path p = Path.of("someText.txt");
        String line = null;
        try (var reader = Files.newBufferedReader(p)) {
            line = reader.readLine();

        } catch (IOException e) {
            e.printStackTrace();
        }
        return line;
    }
}
```

4.

```
public class BadCharException extends RuntimeException {
    private static final long serialVersionUID = 1L;

    public BadCharException(String message) {
        super(message);
    }
}
```

5. D

6. スーパークラスの IOException が、サブクラスの FileNotFoundException よりも先にチャッチされている

Exercise 30　列挙型

1.

```
public enum DrinkType { COFFEE, TEA, JUICE}
```

2.

```
public class Ex30_2{
    public static void main(String[] args) {
        DrinkType selectedDrink = DrinkType.COFFEE;

        switch(selectedDrink){
            caseCOFFEE   ->  System.out.println(”コーヒー系飲料です”);
            caseTEA      ->  System.out.println(”紅茶系飲料です”);
            default      ->  System.out.println(”ジュール類です”);
        }
    }
}
```

3.

```
public enum DrinkType {
    COFFEE {
        @Override
        public String getCaffeineLevel() {
            return ”High”;  // コーヒーはカフェイン多め
        }
    },
    TEA {
        @Override
        public String getCaffeineLevel() {
            return ”Medium”;  // 紅茶は中程度
        }
    },
    JUICE {
        @Override
        public String getCaffeineLevel() {
            return ”None”;  // ジュースは基本的にカフェインなし
        }
    };
   public abstract String getCaffeineLevel();
}
```

4.

```
import java.util.List;
public class Ex30_4{
    public static void main(String[] args) {

        var drinkTypes = List.of(DrinkType.COFFEE, DrinkType.TEA, DrinkType.JUICE);

        drinkTypes.stream()
                  .forEach(t->System.out.println(t.getCaffeineLevel()));

    }
}
```

Exercise 31　正規表現

1.

(1) 4 桁の数字のみからなる文字列にマッチする

```
^[0-9]{4}$
```

(2) 英小文字と数字のみを 1 文字以上含む文字列にマッチする

```
^[a-z0-9]+$
```

(3) 英文字のみの単語 2 つが、スペース 1 つで区切られた文字列にマッチする

```
^[A-Za-z]+¥s[A-Za-z]+$
```

(4) 先頭が a、末尾が c で、間に任意の文字が 0 文字以上続く文字列にマッチする

```
^a.*c$
```

(5) color でも colors てもマッチする

```
^colou?s$
```

(6) 先頭が大文字、その後に英小文字が 1 文字以上続く文字列にマッチする

```
^[A-Z][a-z]+$
```

(7) c と t の間に任意の 1 文字が入る 3 文字の文字列にマッチする

```
^c.t$
```

(8) 年(4 桁)-月(2 桁)-日(2 桁)の形式にマッチする

```
^¥d{4}-¥d{2}-¥d{2}$
```

2.

(1) 数字を最低 1 つ含む全体の長さが 6 文字以上の文字列にマッチする

```
^(?=.*[0-9]).{6,}$
```

(2) 英数字以外を低 1 つ含む全体の長さが 6 文字以上の文字列にマッチする

```
^(?=.*[^A-Za-z0-9]).{6,}$
```

3.

```
import java.util.List;

public class Ex31_3 {
   public static void main(String[] args) {

        var timeList = List.of("12:31","00:00", "24:00", "23:59","12:60");
        String timeRegex = "^([01]¥¥d|2[0-3]):[0-5]¥¥d$";

        // ストリームでフィルタリングし、マッチしたものだけ表示
        timeList.stream()
                .filter(time -> time.matches(timeRegex))
                .forEach(System.out::println);

   }
```

Exercise 32　日付と時刻

1.

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
public class Ex32_1 {
    public static void main(String[] args) {
        System.out.println(LocalDate.of(2025, 3, 1));
        System.out.println(LocalTime.of(23, 5, 45));
        System.out.println(LocalDateTime.of(2025, 3, 1, 23, 5, 45));
    }
}
```

2.

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
public class Ex32_2 {
    public static void main(String[] args) {
        System.out.println(LocalDate.parse("2025-03-01"));
        System.out.println(LocalTime.parse("23:05:45"));
        System.out.println(LocalDateTime.parse("2025-03-01T23:05:45"));
    }
}
```

3.

```
import java.time.LocalDate;
import java.time.chrono.JapaneseDate;
import java.time.format.DateTimeFormatter;

public class Ex32_3 {

    public static void main(String[] args) {
        var format = DateTimeFormatter.ofPattern("Gy 年 M 月 d 日 eeee");
        var jdate = JapaneseDate.from(LocalDate.now());
        System.out.println(jdate.format(format));
    }
}
```

4.

```
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
public class Ex32_4 {
   public static void main(String[] args) {
       var formatter = DateTimeFormatter.ofPattern("ah 時 m 分 s 秒");
       System.out.println(LocalTime.now().format(formatter));
   }
}
```

5.

```
import java.time.LocalDate;
public class Ex32_5 {
   public static void main(String[] args) {
       var date = LocalDate.of(2025, 1, 3)
                   .plusYears(5)
                   .plusMonths(7)
                   .plusDays(13);

       System.out.println(date);
   }
}
```

6.

```
import java.time.LocalDate;
import java.util.List;

public class Ex32_6 {
   public static void main(String[] args) {
       var dates = List.of(
                   LocalDate.of(2021, 3, 20),
                   LocalDate.of(2025, 4, 8),
                   LocalDate.of(2025, 8, 3),
                   LocalDate.of(2024, 11, 13),
                   LocalDate.of(2025, 10, 2));

       var  referenceDate = LocalDate.of(2025, 7, 31);
       dates.stream()
            .filter(d->d.isBefore(referenceDate))
            .forEach(System.out::println);
   }
}
```

7.

```
import java.time.LocalDate;
import java.time.Period;
import java.time.temporal.ChronoUnit;

public class Ex32_7 {

   public static void main(String[] args) {
        LocalDate birthday = LocalDate.of(1979,1,3);
        LocalDate entryDate = LocalDate.of(2002, 4, 1);
       System.out.println(ChronoUnit.YEARS.between(birthday, LocalDate.now()) + "才");

        var p = Period.between(entryDate, LocalDate.now());
        System.out.printf("勤続%d 年%d ケ月%n",p.getYears(), p.getMonths());
   }
}
```

Exercise 33　ファイル入出力

1.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.List;
public class Ex33_1 {

    public static void main(String[] args) {
        var csv = List.of(
                "ID,Name,Age,Email,Phone",
                "1,佐藤 太郎,28,taro.sato@example.com,090-1234-5678",
                "2,鈴木 花子,34,hanako.suzuki@example.com,080-2345-6789",
                "3,木村 翔太,42,shota.kimura@example.com,070-3456-7890",
                "4,山田 佳林,25,karin.yamada@example.com,090-4567-8901",
                "5,小林 春樹,31,haruki.kobayashi@example.com,080-5678-9012");

        Path p = Path.of("meibo.csv");
        try {
            Files.write(p, csv);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
public class Ex33_2 {
    public static void main(String[] args) {

        Path p = Path.of("meibo.csv");
        try {
            var list = Files.readAllLines(p);
            list.forEach(System.out::println);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

3.

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.List;
public class Ex33_3 {
   public static void main(String[] args) {

        var csv = List.of(
                "ID,Name,Age,Email,Phone",
                "1,佐藤 太郎,28,taro.sato@example.com,090-1234-5678",
                "2,鈴木 花子,34,hanako.suzuki@example.com,080-2345-6789",
                "3,木村 翔太,42,shota.kimura@example.com,070-3456-7890",
                "4,山田 佳林,25,karin.yamada@example.com,090-4567-8901",
                "5,小林 春樹,31,haruki.kobayashi@example.com,080-5678-9012");

        String stringPath = "meibo.txt";
        try (var writer = new PrintWriter(stringPath)) {
            for(String s : csv) {
                writer.println(s);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
   }
}
```

4.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
public class Ex33_4 {
   public static void main(String[] args) {

        var p = Path.of("meibo.txt");
        try (var reader = Files.newBufferedReader(p)) {
            String line;
            while((line=reader.readLine())!=null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
   }
}
```

5.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
public class Ex33_5 {
   public static void main(String[] args) {

       try {
           Files.createDirectories(Path.of("meibo/csv"));
           Files.createDirectories(Path.of("meibo/txt"));
       } catch (IOException e) {
           e.printStackTrace();
       }
   }
}
```

6.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
public class Ex33_6 {
   public static void main(String[] args) {

       try {
           Files.move(Path.of("meibo.csv"), Path.of("meibo/csv/meibo.csv"));
           Files.move(Path.of("meibo.txt"), Path.of("meibo/txt/meibo.txt"));
       } catch (IOException e) {
           e.printStackTrace();
       }
   }
}
```

7.

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
public class Ex33_7 {
   public static void main(String[] args) {
       var path = Path.of("meibo/");
       try (var stream = Files.walk(path)) {
           stream.filter(Files::isRegularFile)
                 .map(Path::getFileName)
                 .forEach(System.out::println);
       } catch (IOException e) {
           e.printStackTrace();
       }
   }
}
```

Exercise 34　マルチスレッド　練習問題はなし

Exercise 35-1　ウェブアプリケーション

解答は省略します。

Exercise 35-2　ウェブアプリケーション

2.

```
package jp.kwebs.myapp.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class MyappController {

    @GetMapping(”/myapp”)
    public String display() {
        return ”comment”;
    }
}
```

Exercise 35-3　ウェブアプリケーション

3.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>comment</title>
</head>
<body>
<h1>コメント</h1>
</body>
</html>
```

Exercise 35-4　ウェブアプリケーション

4．Myapp2 プロジェクト

（１）comment-entry.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>comment</title>
</head>
<body>
<h1>コメントの入力</h1>
<form th:action="@{/myapp}" method="post">
コメント：
<input type="text" name="comment">
<input type="submit" value="送信">
</form>
</body>
</html>
```

（２）コントローラー

```
package jp.kwebs.myapp.controller;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class MyappController {

   @GetMapping("/myapp")
   public String display() {
       return "comment-entry";
   }
}
```

Exercise 35-5　ウェブアプリケーション

5. MyappController

```
@Controller
public class MyappController {

    @GetMapping(”/myapp”)
    public String display() {
        return ”comment-entry”;
    }

    @PostMapping(”/myapp”)
    public String commentData(@RequestParam String comment, Model model) {
        model.addAttribute(”comment”, comment);
        return  ”comment-result”;
    }
}
```

Exercise 35-6　ウェブアプリケーション

6. comment-result.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>bookResut</title>
</head>
<body>
   <h1>入力されたコメント</h1>
   コメント：<span th:text="${comment}">なかなかおもしろい！</span>
</body>
</html>
```

7. 省略

Exercise 36-1　フォーム部品　テキストボックス

1.省略

2. TextboxController.java

```
@Controller
public class TextboxController {

   @GetMapping(”/address”)
   public String display() {
       return ”textbox-enter”;
   }

   @PostMapping(”/address”)
   public String result(@RequestParam String address, Model model) {
       model.addAttribute(”address”, address);
       return ”textbox-result”;
   }
}
```

textbox-enter.html

```
<!DOCTYPE html>
<html xmlns:th=”http://www.thymeleaf.org”>
<head>
<meta charset=”UTF-8”>
<title>address</title>
</head>
<body>
   <form th:action=”@{/address}” method=”post”>
       <input type=”text” name=”address”>
       <input type=”submit” value=”送信”>
   </form>
</body>
</html>
```

textbox-result.html

```
<!DOCTYPE html>
<html xmlns:th=”http://www.thymeleaf.org”>
<head>
<meta charset=”UTF-8”>
<title>address</title>
</head>
<body>
住所：
<span th:text=”${address}”>東京都</span>
</body>
</html>
```

Exercise 36-2　フォーム部品 日付の入力

DateController.java

```
@Controller
public class DateController {

    @GetMapping("/holiday")
    public String display() {
        return "date-enter";
    }

    @PostMapping("/holiday")
    public String result(@RequestParam LocalDate holiday, Model model) {
        model.addAttribute("holiday", holiday);
        return "date-result";
    }
}
```

date-enter.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>date</title>
</head>
<body>
    <form th:action="@{/holiday}" method="post">
        <div>
            <input type="date" name="holiday">
            <input type="submit" value="送信">
        </div>
    </form>
</body>
</html>
```

date-result.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>date</title>
</head>
<body>
    祝日:
    <span th:text="${holiday}">2000-01-01</span>
</body>
</html>
```

Exercise 36-3　フォーム部品 ラジオボタン

RadioController.java

```
@Controller
public class RadioController {

   @GetMapping(”/city”)
   public String display() {
       return ”radio-enter”;
   }

   @PostMapping(”/city”)
   public String result(@RequestParam String city, Model model) {
       model.addAttribute(”city”, city);
       return ”radio-result”;
   }
}
```

radio-enter.html

```
<!DOCTYPE html>
<html xmlns:th=”http://www.thymeleaf.org”>
<head>
<meta charset=”UTF-8”>
<title>city</title>
</head>
<body>
   <form th:action=”@{/city}” method=”post”>
       <input type=”radio” name=”city” value=”東京”>東京
       <input type=”radio” name=”city” value=”大阪”>大阪
       <input type=”radio” name=”city” value=”福岡”>福岡
       <br><input type=”submit” value=”送信”>
   </form>
</body>
</html>
```

radio-result.html

```
<!DOCTYPE html>
<html xmlns:th=”http://www.thymeleaf.org”>
<head>
<meta charset=”UTF-8”>
<title>city</title>
</head>
<body>
   都市：
   <span th:text=”${city}”>東京</span>
</body>
</html>
```

Exercise 36-4　フォーム部品 チェックボックス

CheckboxController.java

```
@Controller
public class CheckboxController {

   @GetMapping(”/fruit”)
   public String display() {
       return ”checkbox-enter”;
   }

   @PostMapping(”/fruit”)
   public String result(@RequestParam List<String> fruit, Model model) {
       model.addAttribute(”fruit”, fruit);
       return ”checkbox-result”;
   }
}
```

checkbox-enter.html

```
<!DOCTYPE html>
<html xmlns:th=”http://www.thymeleaf.org”>
<head>
<meta charset=”UTF-8”>
<title>fruit</title>
</head>
<body>
   <form th:action=”@{/fruit}” method=”post”>
       <input type=”checkbox” name=”fruit” value=”リンゴ”>リンゴ
       <input type=”checkbox” name=”fruit” value=”オレンジ”>オレンジ
       <input type=”checkbox” name=”fruit” value=”ぶどう”>ぶどう
       <br><input type=”submit” value=”送信”>
   </form>
</body>
</html>
```

checkbox-result.html

```
<!DOCTYPE html>
<html xmlns:th=”http://www.thymeleaf.org”>
<head>
<meta charset=”UTF-8”>
<title>fruit</title>
</head>
<body>
   好きな果物：<br>
   <span th:text=”${fruit}”>[リンゴ,オレンジ]</span>
</body>
</html>
```

Exercise 36-5　フォーム部品 ドロップダウンリスト

DropdownController.java

```
@Controller
public class DropdownController {

    @GetMapping("/transport")
    public String display() {
         return "dropdown-enter";
    }

    @PostMapping("/transport")
    public String result(@RequestParam String transport, Model model) {
         model.addAttribute("transport", transport);
         return "dropdown-result";
    }

}
```

dropdown-enter.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>select</title>
</head>
<body>
    <form th:action="@{/transport}" method="post">
         <select name="transport">
              <option value="none">選択してください</option>
              <option value="鉄道">鉄道</option>
              <option value="飛行機">飛行機</option>
              <option value="バス">バス</option>
         </select>
         <input type="submit" value="送信">
    </form>
</body>
</html>
```

dropdown-result.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>select</title>
</head>
<body>
交通：
<span th:text="${transport}">東京</span>
</body>
</html>
```

Exercise 36-6　フォーム部品　テキストエリア

TextareaController.java

```
@Controller
public class TextareaController {

    @GetMapping(”/notice”)
    public String display() {
         return ”textarea-enter”;
    }

    @PostMapping(”/notice”)
    public String textareaResult(@RequestParam String notice, Model model) {
         model.addAttribute(”notice”, notice);
         return ”textarea-result”;
    }
}
```

Textarea-enter.html

```
<!DOCTYPE html>
<html xmlns:th=”http://www.thymeleaf.org”>
<head>
<meta charset=”UTF-8”>
<title>text area</title>
</head>
<body>
    <form th:action=”@{/notice}” method=”post”>
         お知らせ：<br>
         <textarea name=”notice” rows=”4” cols=”50”></textarea>
         <br><input type=”submit” value=”送信”>
    </form>
</body>
</html>
```

textarea-result.html

```
<!DOCTYPE html>
<html xmlns:th=”http://www.thymeleaf.org”>
<head>
<meta charset=”UTF-8”>
<title>text area</title>
</head>
<body>
    お知らせ:<br>
    <div th:text=”${notice}” style=”white-space: pre;”>
    １行目<br>
    ２行目
    </div>
</body>
</html>
```

Exercise 37-1　Thymeleaf: 繰り返しと条件付き表示

1．省略

2．リストのデータをウェブに表示する処理

(1) コントローラーメソッド(exercise_371)

```
    @GetMapping("/ex371")
    public String exercise_371(Model model) {
        var items = List.of("apple", "banana", "cherry");
        model.addAttribute("items", items);
        model.addAttribute("stock", true);
        return "/ex371";
    }
```

(2)ex371.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
    <ul th:if="${stock}">
        <li th:each="item : ${items}" th:text="${item}">grape</li>
    </ul>
    <p th:unless="${stock}">在庫なし</p>
</body>
</html>
```

Exercise 37-2 Thymeleaf: オブジェクトの表示

1.Bread.java

```
public class Bread {
    private String name;
    private int price;

    public Bread() {
    }
    public Bread(String name, int price) {
        this.name = name;
        this.price = price;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
        this.price = price;
    }
}
```

2.コントローラーメソッド（exercise_372）

```
    @GetMapping("/ex372")
    public String exercise_372(Model model) {
        var bread = new Bread("フランスパン", 350);
        model.addAttribute("bread", bread);
        return "ex372";
    }
```

3.ex372.html

```
 <!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
    <div th:object="${bread}" >
        品名：<span th:text="*{name}">アンパン</span><br>
        価格：<span th:text="*{price}">150</span>
    </div>
</body>
</html>
```

Exercise 37-3 Thymeleaf: オブジェクトとして受け取る

1.コントローラーメソッド

```
    @GetMapping("/ex373")
    public String exercise_373(Model model) {
        return "ex373";
    }

    @PostMapping("/ex373")
    public String exercise_373_result(Bread bread) {
        return "ex372"; // 前回作ったものをそのまま利用
    }
```

2. ex373.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title></title>
</head>

<body>
    <form th:action="@{/ex373}" method="post">
        品名:<input type="text" name="name"><br>
        価格:<input type="text" name="price"><br>
        <input type="submit" value="送信">
    </form>
</body>

</html>
```

Exercise 37-4 Thymeleaf: フォームとオブジェクトをバインドする

1.コントローラーメソッド

```
    @GetMapping("/ex374")
    public String exercise_374(Model model) {
        model.addAttribute("bread", new Bread("フランスパン", 350));
        return "ex374";
    }

    @PostMapping("/ex374")
    public String exercise_374_result(Bread bread) {
        return "ex372";// 前回作ったものをそのまま利用
    }
```

テンプレート（ex374.html）

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title></title>
</head>

<body>
    <form th:action="@{/ex374}" method="post" th:object="${bread}">
        品名:<input type="text" th:field="*{name}"><br>
        価格:<input type="text" th:field="*{price}"><br>
        <input type="submit" value="送信">
    </form>
</body>

</html>
```

Exercise 38-1 総合演習

1. 省略

Exercise 38-2 総合演習

PcForm.java

```
package jp.kwebs.pcshop.form;
import java.util.List;
import jakarta.validation.constraints.Max;
import jakarta.validation.constraints.Min;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotEmpty;
import jakarta.validation.constraints.NotNull;

public class PcForm {
    private Long id;
    private String cpu;
    private Integer memory;
    private String os;
    private List<String> storage;

    public PcForm() {}
    public PcForm(Long id, String cpu, Integer memory, String os, List<String> storage) {
         this.id = id;
         this.cpu = cpu;
         this.memory = memory;
         this.os = os;
         this.storage = storage;
    }
    public Long getId() {
         return id;
    }
    public String getCpu() {
         return cpu;
    }
    public Integer getMemory() {
         return memory;
    }
    public String getOs() {
         return os;
    }
    public List<String> getStorage() {
         return storage;
    }
    public void setId(Long id) {
         this.id = id;
    }
    public void setCpu(String cpu) {
         this.cpu = cpu;
    }
    public void setMemory(Integer memory) {
         this.memory = memory;
    }
    public void setOs(String os) {
         this.os = os;
    }
    public void setStorage(List<String> storage) {
         this.storage = storage;
    }
}
```

Exercise 38-3 総合演習

PcController.java

```
package jp.kwebs.pcshop.controller;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import jp.kwebs.pcshop.form.PcForm;

@Controller
@RequestMapping(”/pc”)
public class PcController {

    @GetMapping(”/create”)
    public String create(Model model) {
        model.addAttribute(”pcForm”, new PcForm());
        return ”pc-create”;
    }
}
```

Exercise 38-4 総合演習

pc-create.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>PC 仕様入力フォーム</title>
</head>
<body>
    <h2>PC 仕様入力フォーム</h2>
    <form th:action="@{/pc/create}" method="post" th:object="${pcForm}">
        <table>
            <tr>
                <td>CPU:</td>
                <td>
                  <input type="text" th:field="*{cpu}">
                </td>
            </tr>
            <tr>
                <td>メモリ:</td>
                <td>
                  <input type="number" th:field="*{memory}">
                </td>
            </tr>
            <tr>
                <td>OS:</td>
                <td>
                    <select th:field="*{os}">
                        <option value="Windows">Windows</option>
                        <option value="MacOS">MacOS</option>
                        <option value="Linux">Linux</option>
                    </select>
                </td>
            </tr>
            <tr>
                <td>ストレージ:</td>
                <td>
                    <input type="checkbox" th:field="*{storage}" value="HDD">HDD
                    <input type="checkbox" th:field="*{storage}" value="STA-SSD">STA SSD
                    <input type="checkbox" th:field="*{storage}" value="NVMe-SSD">NVMe SSD
                </td>
            </tr>
        </table>
        <input type="submit" value="作成">
    </form>
</body>
</html>
```

Exercise 38-5 総合演習

コントローラーメソッド

```
    @PostMapping("/create")
    public String result(PcForm pc) {
        return "pc-result";
    }
```

pc-result.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>PC の仕様</title>
</head>
<body>
    <h1>PC の仕様</h1>
    <table th:object="${pcForm}">
        <tr>
            <td>CPU：</td>
            <td th:text="*{cpu}">Intel</td>
        </tr>
        <tr>
            <td>メモリ：</td>
            <td th:text="*{memory}+'GB'">16GB</td>
        </tr>
        <tr>
            <td>OS：</td>
            <td th:text="*{os}">Windows</td>
        </tr>
        <tr>
            <td>ストレージ：</td>
            <td th:text="*{storage}">[HDD,STA-SSD]</td>
        </tr>
    </table>
</body>
</html>
```

Exercise 38-6 総合演習

pc-result.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>PC の仕様</title>
</head>
<body>
    <h1>PC の仕様</h1>
    <table th:object="${pcForm}">
        <tr>
            <td>CPU：</td>
            <td th:text="*{cpu}">Intel</td>
        </tr>
        <tr>
            <td>メモリ：</td>
            <td th:text="*{memory}+'GB'">16GB</td>
        </tr>
        <tr>
            <td>OS：</td>
            <td th:text="*{os}">Windows</td>
        </tr>
        <!--
        <tr>
            <td>ストレージ：</td>
            <td th:text="*{storage}">[HDD,STA-SSD]</td>
        </tr>
         -->
        <tr>
            <td>ストレージ：</td>
            <td th:text="${#strings.listJoin(pcForm.storage, ',')}">HDD,STA-SSD</td>
        <tr>
    </table>
</body>
</html>
```

Exercise 39-1 バリデーション　バリデータ

1. 省略

2.

```
import java.util.List;

import jakarta.validation.constraints.Max;
import jakarta.validation.constraints.Min;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotEmpty;
import jakarta.validation.constraints.NotNull;

public class PcFormV {
    private Long id;
    @NotBlank
    private String cpu;
    @NotNull
    @Max(value=128)
    @Min(value=4)
    private Integer memory;
    @NotNul
    private String os;
    @NotEmpty
    private List<String> storage;

    public PcFormV() {}
    public PcFormV(Long id, String cpu, Integer memory, String os, List<String> storage) {
         this.id = id;
         this.cpu = cpu;
         this.memory = memory;
         this.os = os;
         this.storage = storage;
    }

    // ゲッターとセッターは記載を省略


}
```

Exercise 39-2 バリデーション　バリデーションの実行

コントローラーメソッド（バリデーションを実行する）

```
    @PostMapping(”/create”)
    public String result(@Valid PcForm pc, BindingResult result) {
        if(result.hasErrors()) {
            return ”pc-create”;
        }
        return ”pc-result”;
    }
```

Exercise 39-3 バリデーション　エラーメッセージの表示

1.PcForm にバリデーションメッセージを追加

```
import java.util.List;

import jakarta.validation.constraints.Max;
import jakarta.validation.constraints.Min;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotEmpty;
import jakarta.validation.constraints.NotNull;

public class PcFormV {
    private Long id;
    @NotBlank(message="CPU を記入してください")
    private String cpu;
    @NotNull(message="メモリを記入してください")
    @Max(value=128, message="128GB 以内です")
    @Min(value=4, message="4GB 以上です")
    private Integer memory;
    @NotNull(message="OS を選択してください")
    private String os;
    @NotEmpty(message="1 つ以上選択してください")
    private List<String> storage;

    public PcFormV() {}
    public PcFormV(Long id, String cpu, Integer memory, String os, List<String> storage) {
         this.id = id;
         this.cpu = cpu;
         this.memory = memory;
         this.os = os;
         this.storage = storage;
    }

    // ゲッターとセッターは記載を省略


}
```

2.バリデーションメッセージを表示する pc-create.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>PC 仕様入力フォーム</title>
   <style>
   .msg{
       color: red;
       font-size: 0.8em;
   }
   </style>
</head>
<body>
    <h2>PC 仕様入力フォーム</h2>
    <form th:action="@{/pc/create}" method="post" th:object="${pcForm}">
        <table>
            <tr>
                <td>CPU:</td>
                <td>
                  <input type="text" th:field="*{cpu}">
                  <span th:errors="*{cpu}" class="msg"></span>
                </td>
            </tr>
            <tr>
                <td>メモリ:</td>
                <td>
                  <input type="number" th:field="*{memory}">
                  <span th:errors="*{memory}" class="msg"></span>
                </td>
            </tr>
            <tr>
                <td>OS:</td>
                <td>
                    <select th:field="*{os}">
                        <option value="Windows">Windows</option>
                        <option value="MacOS">MacOS</option>
                        <option value="Linux">Linux</option>
                    </select>
                      <span th:errors="*{os}" class="msg"></span>
                </td>
            </tr>
            <tr>
                <td>ストレージ:</td>
                <td>
                    <input type="checkbox" th:field="*{storage}" value="HDD">HDD
                    <input type="checkbox" th:field="*{storage}" value="STA-SSD">STA SSD
                    <input type="checkbox" th:field="*{storage}" value="NVMe-SSD">NVMe SSD
                    <span th:errors="*{storage}" class="msg"></span>
                </td>
            </tr>
        </table>
        <input type="submit" value="作成">
    </form>
</body>
</html>
```

Exercise 40-1　データベースプロジェクトの作成

解答を省略します

Exercise 40-2　データベースの設定

解答を省略します

Exercise 40-3　エンティティの作成

Pc.java

```
import java.util.List;
import jakarta.persistence.ElementCollection;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;

@Entity
public class Pc {
    @Id
    @GeneratedValue
    private Long id;
    private String cpu;
    private Integer memory;
    private String os;
    @ElementCollection
    private List<String> storage;

    public Pc() {}
    public Pc(Long id, String cpu, Integer memory, String os, List<String> storage) {
        this.id = id;
        this.cpu = cpu;
        this.memory = memory;
        this.os = os;
        this.storage = storage;
    }

    // ゲッターとセッターは記載を省略

}
```

Exercise 40-4　リポジトリの作成

PcRepository.java

```
import org.springframework.data.jpa.repository.JpaRepository;
import jp.kwebs.pcshop.entiry.Pc;

public interface PcRepository extends JpaRepository<Pc, Long> {

}
```

Exercise 40-5　サービスクラスの作成 I

PcService.java

```
import java.util.List;
import org.springframework.stereotype.Service;
import jakarta.transaction.Transactional;
import jp.kwebs.pcshop.entiry.Pc;
import jp.kwebs.pcshop.form.PcForm;
import jp.kwebs.pcshop.repository.PcRepository;

@Service
public class PcService {

    private final PcRepository repo;

    public PcService(PcRepository repo) {
        this.repo = repo;
    }

}
```

Exercise 40-6　サービスクラスの作成２

PcService.java

```
import java.util.List;
import org.springframework.stereotype.Service;
import jakarta.transaction.Transactional;
import jp.kwebs.pcshop.entiry.Pc;
import jp.kwebs.pcshop.form.PcForm;
import jp.kwebs.pcshop.repository.PcRepository;

@Service
public class PcService {

    private final PcRepository repo;

    public PcService(PcRepository repo) {
        this.repo = repo;
    }

    public List<Pc> readAllPcs() {
        return repo.findAll();
    }
    public void createAllPcs(List<Pc> pcs) {
        repo.saveAll(pcs);
    }
    public Pc readPcById(Long id) {
        return repo.findById(id).orElseThrow();
    }
    @Transactional
    public void createPc(PcForm pcForm) {
        Pc pc = new Pc();
        toEntity(pc, pcForm);
        repo.save(pc);
    }
    @Transactional
    public Pc updatePc(PcForm pcForm) {
        Pc pc = readPcById(pcForm.getId());
        toEntity(pc, pcForm);
        repo.save(pc);
        return pc;
    }
    @Transactional
    public void deletePc(Long id) {
        repo.deleteById(id);
    }
    public void toEntity(Pc pc, PcForm pcForm) {
        pc.setCpu(pcForm.getCpu());
        pc.setMemory(pcForm.getMemory());
        pc.setOs(pcForm.getOs());
        pc.setStorage(pcForm.getStorage());

    }
    public PcForm toForm(Pc pc) {
        return new PcForm(pc.getId(), pc.getCpu(), pc.getMemory(), pc.getOs(), pc.getStorage());
    }
}
```

Exercise 40-7　データベースの初期化

1. Util.java （リソースからダウンロードしてください）

```
package jp.kwebs.pcshop.util;
import java.util.List;
import jp.kwebs.pcshop.entiry.Pc;

public class Util {
    public static List<Pc> getPcs() {
        var list = List.of(
             new Pc(null, "Core i7", 16, "Windows", List.of("HDD","STA-SSD")),
             new Pc(null, "Core i5", 8, "Windows", List.of("STA-SSD")),
             new Pc(null, "M3", 4, "MacOS", List.of("STA-SSD")),
             new Pc(null, "Core i9", 32, "Windows", List.of("HDD", "STA-SSD")),
             new Pc(null, "Core i7", 16, "Windows", List.of("HDD", "STA-SSD")),
             new Pc(null, "M3", 8, "MacOS", List.of("STA-SSD")),
             new Pc(null, "M3", 4, "MacOS", List.of("HDD")),
             new Pc(null, "Core i9", 32, "Windows", List.of("HDD", "STA-SSD", "NVMe-SSD")),
             new Pc(null, "M2", 16, "MacOS", List.of("HDD")),
             new Pc(null, "Core i5", 8, "Linux", List.of("STA-SSD")),
             new Pc(null, "Core i3", 4, "Windows", List.of("HDD")),
             new Pc(null, "Core i9", 32, "Windows", List.of("HDD", "STA-SSD")),
             new Pc(null, "Core i7", 16, "Windows", List.of("HDD", "STA-SSD")),
             new Pc(null, "M2", 8, "MacOS", List.of("STA-SSD")),
             new Pc(null, "Core i3", 4, "Windows", List.of("STA-SSD")) );

        return list;
    }
}
```

2. DatabaseInitializer.java

```
package jp.kwebs.pcshop.config;
import org.springframework.boot.CommandLineRunner;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import jp.kwebs.pcshop.service.PcService;
import jp.kwebs.pcshop.util.Util;

@Configuration
public class DatabaseInitializer {

   @Bean
   CommandLineRunner init(PcService ps) {
        CommandLineRunner clr = s->ps.createAllPcs(Util.getPcs());
        return clr;
   }
}
```

Exercise 41-1　アプリケーションの作成

(1) PcController.java

```
@Controller
@RequestMapping(”/pc”)
public class PcController {

    private final PcService ps;

    public PcController(PcService ps) {
        this.ps = ps;
    }

    @GetMapping(”/list”)
    public String listing(Model model) {

        var pcs = ps.readAllPcs();
        model.addAttribute(”pcs”, pcs);
        return ”pc-list”;
    }
}
```

Exercise 41-2　アプリケーションの作成

pc-list.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Book List</title>
</head>
<body>
    <h1>PC リスト</h1>
    <table>
        <tr>
            <th>ID</th>
            <th>CPU</th>
            <th>メモリー</th>
            <th>OS</th>
            <th>ストレージ</th>
        </tr>
        <tr th:each="pc : ${pcs}">
            <td th:text="${pc.id}">1</td>
            <td th:text="${pc.cpu}">Core i9</td>
            <td th:text="${pc.memory}">1128</td>
            <td th:text="${pc.os}">Windows</td>
            <td th:text="${#strings.listJoin(pc.storage, ',')}"></td>
            <td>
                <a th:href="@{/pc/{id}/edit   (id=${pc.id}) }">修正</a>
                <a th:href="@{/pc/{id}/delete (id=${pc.id}) }">削除</a>
            </td>
        </tr>
    </table>
    <a th:href="@{/pc/create}">新規作成</a>

</body>
</html>
```

Exercise 41-3　アプリケーションの作成

(1)、(2)

```
@Controller
@RequestMapping("/pc")
public class PcController {

    private final PcService ps;

    public PcController(PcService ps) {
        this.ps = ps;
    }

    @GetMapping("/list")
    public String listing(Model model) {

        var pcs = ps.readAllPcs();
        model.addAttribute("pcs", pcs);
        return "pc-list";
    }

    @GetMapping("/create")
    public String pcForm(Model model) {
        model.addAttribute("pcForm", new PcForm());
        return "pc-create";
    }

    @PostMapping("/create")
    public String create(@Valid PcForm pcForm, BindingResult result) {
        if (result.hasErrors()) {
            return "pc-create";
        }
        ps.createPc(pcForm);
        return "redirect:/pc/list";
    }

}
```

Exercise 41-4　アプリケーションの作成

(1)

```
@Controller
@RequestMapping(”/pc”)
public class PcController {

    private final PcService ps;

    public PcController(PcService ps) {
        this.ps = ps;
    }

    @GetMapping(”/list”)
    public String listing(Model model) {

        var pcs = ps.readAllPcs();
        model.addAttribute(”pcs”, pcs);
        return ”pc-list”;
    }

    @GetMapping(”/create”)
    public String pcForm(Model model) {
        model.addAttribute(”pcForm”, new PcForm());
        return ”pc-create”;
    }

    @PostMapping(”/create”)
    public String create(@Valid PcForm pcForm, BindingResult result) {
        if (result.hasErrors()) {
            return ”pc-create”;
        }
        ps.createPc(pcForm);
        return ”redirect:/pc/list”;
    }

    @GetMapping(”/{id}/edit”)
    public String edit(@PathVariable Long id, Model model) {
        var pc = ps.readPcById(id);
        var pcForm = ps.toForm(pc);
        model.addAttribute(”pcForm”, pcForm);
        return ”pc-edit”;
    }

}
```

(2) pc-edit.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>PC 仕様入力フォーム</title>
   <style>
   .msg{
       color: red;
       font-size: 0.8em;
   }
   </style>
</head>
<body>
    <h2>PC 仕様入力フォーム</h2>
    <form th:action="@{/pc/edit}" method="post" th:object="${pcForm}">
        <table>
            <tr>
                <td>CPU:</td>
                <td>
                  <input type="text" th:field="*{cpu}">
                  <span th:errors="*{cpu}" class="msg"></span>
                </td>
            </tr>
            <tr>
                <td>メモリ:</td>
                <td>
                  <input type="number" th:field="*{memory}">
                  <span th:errors="*{memory}" class="msg"></span>
                </td>
            </tr>
            <tr>
                <td>OS:</td>
                <td>
                    <select th:field="*{os}">
                        <option value="">選択してください</option>
                        <option value="Windows">Windows</option>
                        <option value="MacOS">MacOS</option>
                        <option value="Linux">Linux</option>
                    </select>
                      <span th:errors="*{os}" class="msg"></span>
                </td>
            </tr>
            <tr>
                <td>ストレージ:</td>
                <td>
                    <input type="checkbox" th:field="*{storage}" value="HDD">HDD
                    <input type="checkbox" th:field="*{storage}" value="STA-SSD">STA SSD
                    <input type="checkbox" th:field="*{storage}" value="NVMe-SSD">NVMe SSD
                    <span th:errors="*{storage}" class="msg"></span>
                </td>
            </tr>
        </table>
        <input type="submit" value="修正">
        <input type="hidden" th:field="*{id}">
    </form>
</body>
</html>
```

Exercise 41-5　アプリケーションの作成

updateメソッド

```
@Controller
@RequestMapping(”/pc”)
public class PcController {

    private final PcService ps;

    public PcController(PcService ps) {
        this.ps = ps;
    }

    @GetMapping(”/list”)
    public String listing(Model model) {

        var pcs = ps.readAllPcs();
        model.addAttribute(”pcs”, pcs);
        return ”pc-list”;
    }

    @GetMapping(”/create”)
    public String pcForm(Model model) {
        model.addAttribute(”pcForm”, new PcForm());
        return ”pc-create”;
    }

    @PostMapping(”/create”)
    public String create(@Valid PcForm pcForm, BindingResult result) {
        if (result.hasErrors()) {
            return ”pc-create”;
        }
        ps.createPc(pcForm);
        return ”redirect:/pc/list”;
    }

    @GetMapping(”/{id}/edit”)
    public String edit(@PathVariable Long id, Model model) {
        var pc = ps.readPcById(id);
        var pcForm = ps.toForm(pc);
        model.addAttribute(”pcForm”, pcForm);
        return ”pc-edit”;
    }

    @PostMapping(”/edit”)
    public String update(@Valid PcForm pcForm, BindingResult result) {
        if (result.hasErrors()) {
            return ”pc-edit”;
        }
        ps.updatePc(pcForm);
        return ”redirect:/pc/list”;
    }

}
```

Exercise 41-6　アプリケーションの作成

deleteメソッド

```
@Controller
@RequestMapping(”/pc”)
public class PcController {

    private final PcService ps;

    public PcController(PcService ps) {
        this.ps = ps;
    }

    @GetMapping(”/list”)
    public String listing(Model model) {

        var pcs = ps.readAllPcs();
        model.addAttribute(”pcs”, pcs);
        return ”pc-list”;
    }
    @GetMapping(”/create”)
    public String pcForm(Model model) {
        model.addAttribute(”pcForm”, new PcForm());
        return ”pc-create”;
    }
    @PostMapping(”/create”)
    public String create(@Valid PcForm pcForm, BindingResult result) {
        if (result.hasErrors()) {
            return ”pc-create”;
        }
        ps.createPc(pcForm);
        return ”redirect:/pc/list”;
    }
    @GetMapping(”/{id}/edit”)
    public String edit(@PathVariable Long id, Model model) {
        var pc = ps.readPcById(id);
        var pcForm = ps.toForm(pc);
        model.addAttribute(”pcForm”, pcForm);
        return ”pc-edit”;
    }
    @PostMapping(”/edit”)
    public String update(@Valid PcForm pcForm, BindingResult result) {
        if (result.hasErrors()) {
            return ”pc-edit”;
        }
        ps.updatePc(pcForm);
        return ”redirect:/pc/list”;
    }

    @GetMapping(”/{id}/delete”)
    public String delete(@PathVariable Long id) {
        ps.deletePc(id);
        return ”redirect:/pc/list”;
    }

}
```