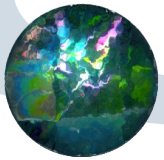
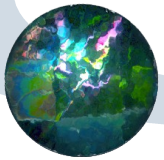
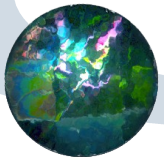


Javaの活用



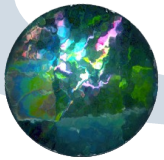
目的

- Javaの様々な機能を使う
- スマートなプログラム開発



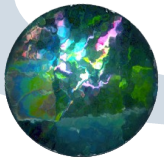
package

- プログラム全体をモジュールに分ける
- 関連の強いクラスをpackageにまとめる
- ソースファイルのフォルダに対応
  - 階層化できる
- ソースファイル先頭でpackage宣言
- 修飾子（public, private)が無いと、package内に公開



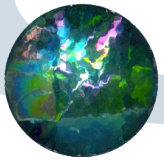
import

- 使用するライブラリクラスの名前解決を支援
- 特定のクラス：import クラス名
  - 例：import java.util.List;
- パッケージ: import パッケージ名
  - 例: import java.awt.*;



修飾子

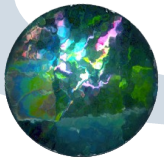
- アクセス制限：public, private, protected
- 定数：final
- クラスに属する変数、メソッド：static
  - クラスインスタンスを作らなくても存在



例外処理

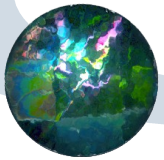
- 実行時に発生するエラーへの対処を記述する
  - ファイルが開かない
  - 正しく値を変換できない
- メソッドが例外を返す：throws
- メソッドの呼出側がそれを処理する

```
try{
  処理
} catch(例外 ex){
  例外処理
}
```



型パラメタの利用（generic)

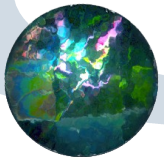
- クラスが保持するデータの型を明示する
  - コンパイル時に整合性を確認する
  - コレクションライブラリ（リンクなど）で重要



Boxing

- 原始型：int, double, boolean, charなど
- 対応するクラス：Integer, Double, Boolean, Character
- 自動でキャストできる

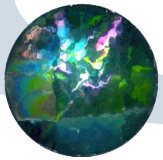
```
int n=Integer.valueOf(10);
Integer nInt=n;
```



拡張されたfor

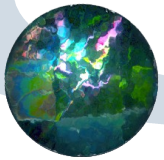
- コレクションの全要素に対するfor

```
for(要素 e : コレクション){
  処理
}
```



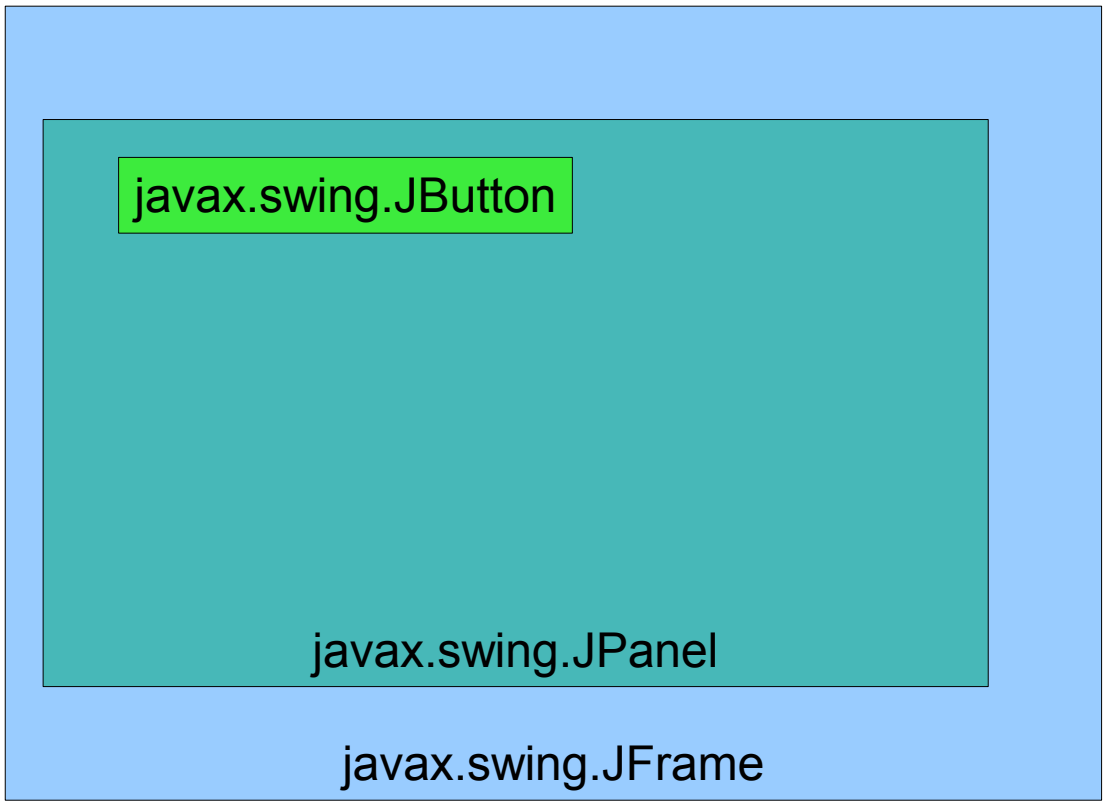
例

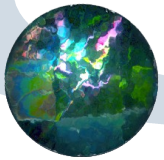
```
java.util.ArrayList<Integer> integers=new java.util.ArrayList<Integer>(
integers.add(10);
integers.add(20);

for(Integer k : integers){
  int n=k;
}
Integer first = integers.get(0);
```



Graphical User Interface

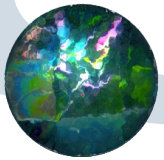
- javaでは、標準にGUIコンポーネントがある
- 基本的コンポーネント：javax.swing.*
- 描画用クラス
  - java.awt.*;
  - java.awt.geom.*;
  - java.awt.Graphics.*;
  - java.awt.Graphics2D.*;


javax.swing.JButton
javax.swing.JPanel
javax.swing.JFrame



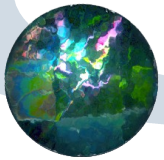
例題2

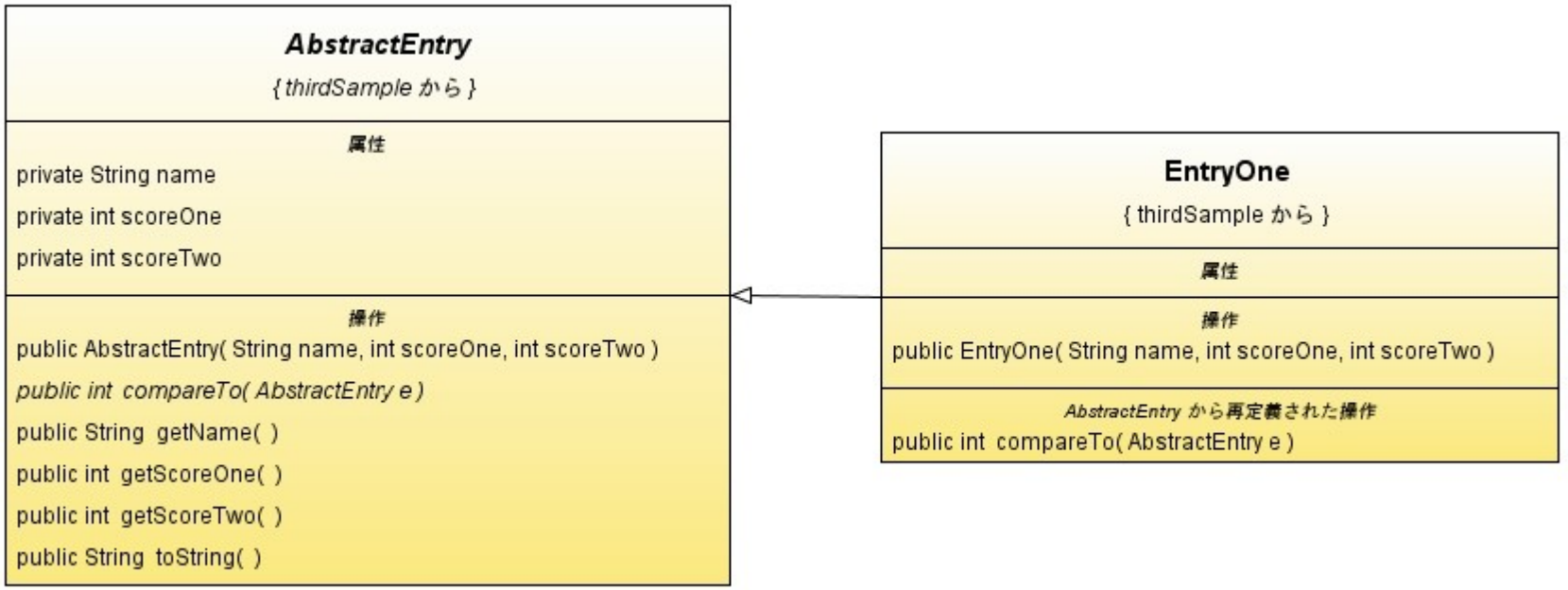
- Entryクラスのインスタンスのリストの保持
  - java.util.ArrayListクラスの利用
  - コレクションとテンプレートの実例
- 二分木によるソートをするクラス
  - java.util.TreeSetクラスの利用
  - java.lang.Comparableインターフェイスによる「自然な順序」



例題3

- scoreを二つ持っていて、どちらでソートするか
- AbstractEntryクラス：抽象クラス
  - compareTo()メソッドを実装せず
- AbstractEntryクラスの継承クラスEntryOne




AbstractEntry
{ thirdSample から }
属性
private String name
private int scoreOne
private int scoreTwo
操作
public AbstractEntry(String name, int scoreOne, int scoreTwo)
public int compareTo(AbstractEntry e)
public String getName( )
public int getScoreOne( )
public int getScoreTwo( )
public String toString( )
EntryOne
{ thirdSample から }
属性
操作
public EntryOne(String name, int scoreOne, int scoreTwo)
AbstractEntry から再定義された操作
public int compareTo(AbstractEntry e)

WithClassNew.java

```
package secondSample;

import java.util.ArrayList;
import java.util.List;
import java.util.TreeSet;
/**
 *
 * @author tadaki
 */
public class WithClassNew<T extends Comparable<T>> {
    //クラスT のインスタンスを要素とするベクトル
    private List<T> entries;

    public WithClassNew(List<T> entries) {
        this.entries = entries;
    }

    public String sort() {
        //クラスT のインスタンスを要素とするソートされた集合
        //ソートは、クラスT のcompareTo()を使って行われる
        TreeSet<T> treeSet = new TreeSet<T>(entries);
        return data2String(treeSet);
    }

    private String data2String(TreeSet<T> treeSet) {
        //配列内の数値を文字列化する
        String nl = System.getProperty("line.separator");

        StringBuilder buffer = new StringBuilder();
        for(T t:treeSet){//拡張されたfor
            buffer.append(t.toString());
            buffer.append(nl);
        }
        return buffer.toString();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        List<EntryNew> entries=new ArrayList<EntryNew>();

        entries.add(new EntryNew("Bob", 90));
        entries.add(new EntryNew("Mary", 70));
```

```
        entries.add(new EntryNew("Tom", 95));
        entries.add(new EntryNew("Mark", 85));
        entries.add(new EntryNew("Betty", 80));
        entries.add(new EntryNew("Tim", 70));
        entries.add(new EntryNew("Ann", 85));
        entries.add(new EntryNew("Kim", 95));

        WithClassNew<EntryNew> withClass =
                new WithClassNew<EntryNew>(entries);

        System.out.println(withClass.sort());
    }
}
```

EntryNew.java

```java
package secondSample;

/**
 *
 * @author tadaki
 */
public class EntryNew implements Comparable<EntryNew> {

    private String name;
    private int score;

    public EntryNew(String name, int score) {
        this.name = name;
        this.score = score;
    }

    public int compareTo(EntryNew e) {
        if (e.getScore() > score) {
            return -1;
        }
        if (e.getScore() < score) {
            return 1;
        }
        return getName().compareTo(e.getName());
    }

    public String getName() {
        return name;
    }

    public int getScore() {
        return score;
    }

    @Override
    public String toString() {
        return getName() + ":" + getScore();
    }
}
```

WithAbstractEntry.java

```
package thirdSample;

import java.util.ArrayList;
import java.util.List;
import java.util.TreeSet;
/**
 *
 * @author tadaki
 */
public class WithAbstractEntry<T extends Comparable<T>> {
    //クラスT のインスタンスを要素とするベクトル
    private List<T> entries;

    public WithAbstractEntry(List<T> entries) {
        this.entries = entries;
    }

    public String sort() {
        //クラスT のインスタンスを要素とするソートされた集合
        //ソートは、クラスT のcompareTo()を使って行われる
        TreeSet<T> treeSet = new TreeSet<T>(entries);
        return data2String(treeSet);
    }

    private String data2String(TreeSet<T> treeSet) {
        //配列内の数値を文字列化する
        String nl = System.getProperty("line.separator");

        StringBuilder buffer = new StringBuilder();
        for(T t:treeSet){//拡張されたfor
            buffer.append(t.toString());
            buffer.append(nl);
        }
        return buffer.toString();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        List<AbstractEntry> entries=new ArrayList<AbstractEntry>();

        entries.add(new EntryOne("Bob", 90,80));
        entries.add(new EntryOne("Mary", 70,60));
```

```
        entries.add(new EntryOne("Tom", 95,80));
        entries.add(new EntryOne("Mark", 85,90));
        entries.add(new EntryOne("Betty", 80,95));
        entries.add(new EntryOne("Tim", 70,80));
        entries.add(new EntryOne("Ann", 85,85));
        entries.add(new EntryOne("Kim", 95,70));

        WithAbstractEntry<AbstractEntry> withClass =
                new WithAbstractEntry<AbstractEntry>(entries);

        System.out.println(withClass.sort());
    }
}
```

AbstractEntry.java

```java
package thirdSample;

/**
 *
 * @author tadaki
 */
public abstract class AbstractEntry
    implements Comparable<AbstractEntry> {

    private String name;
    private int scoreOne;
    private int scoreTwo;

    public AbstractEntry(String name, int scoreOne, int scoreTwo) {
        this.name = name;
        this.scoreOne = scoreOne;
        this.scoreTwo = scoreTwo;
    }

    public abstract int compareTo(AbstractEntry e);

    public String getName() {
        return name;
    }

    public int getScoreOne() {
        return scoreOne;
    }

    public int getScoreTwo() {
        return scoreTwo;
    }

    @Override
    public String toString() {
        return getName() + ":" + getScoreOne() + "," + getScoreTwo();
    }
}
```

EntryOne.java

```java
package thirdSample;

/**
 *
 * @author tadaki
 */
public class EntryOne extends AbstractEntry {

    public EntryOne(String name,  int scoreOne,int scoreTwo) {
        super(name,  scoreOne,scoreTwo);
    }

    public int compareTo(AbstractEntry e) {
        if (e.getScoreOne() > getScoreOne()) {
            return -1;
        }
        if (e.getScoreOne() < getScoreOne()) {
            return 1;
        }
        return getName().compareTo(e.getName());
    }
}
```

EntryTwo.java

```java
package thirdSample;

/**
 *
 * @author tadaki
 */
public class EntryTwo extends AbstractEntry {

    public EntryTwo(String name,  int scoreOne,int scoreTwo) {
        super(name,  scoreOne,scoreTwo);
    }

    public int compareTo(AbstractEntry e) {
        if (e.getScoreTwo() > getScoreTwo()) {
            return -1;
        }
        if (e.getScoreTwo() < getScoreTwo()) {
            return 1;
        }
        return getName().compareTo(e.getName());
    }
}
```