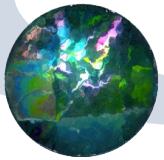
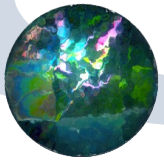
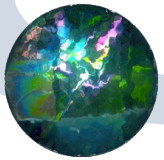


Java入門

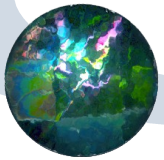


オブジェクト指向プログラミング

- グラフの構造を表すデータ構造
  - グラフ、点、枝
- データが階層的
  - グラフの要素としての点と枝
  - 点に接続している枝のリスト
  - 枝の両端の点

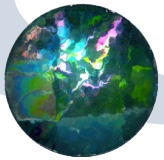


- 各データごとの操作
  - 枝に値を設定する
  - 探索問題：点や枝に印を付ける
- データのカプセル化
  - グラフとしての整合性を維持して、点や枝の性質を変える
- 型の継承と拡張
  - 枝に「流れ」の属性を付けて拡張
  - グラフの可視化



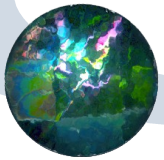
C++ではなくJavaを使う理由

- 豊富なユーティリティー
  - java.util.Vector
- 使い易い開発環境(IDE)
  - NetBeans、Eclipse
- 多数のOSで使える
  - Windows、Linux、Solaris
- GUI開発ができる
  - IDEを使うと簡単

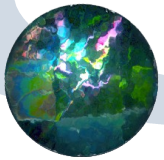


Javaの基本

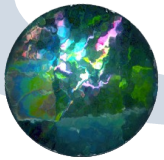
- 全てがクラス
- 開始点となるクラスが必要
  - public static void main(String[] args)メソッドから始まる
  - mainは主となるクラスを起動するだけ
- コンストラクタメソッド
  - クラスと同じ名前のメソッド
- デストラクタは無い
  - 自動ガベージコレクション



- 一つのクラスで一つのファイルが基本
  - ファイル名はクラス名と同じ
- ヘッダファイルが無い
  - ライブラリはimport文を使う
- C/C++のポインタは無い
  - 原始型は値代入
  - クラスオブジェクトは参照

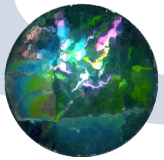


- 文法はだいたいC++と同じ
- 原始型はint、double、char、booleanなど
- 原始型に対応したクラスがある
  - Integer、Double、Character、Booleanなど
- 文字列Stringや原始型の配列はクラス



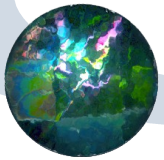
便利なライブラリ

- オンラインマニュアル
- http://java.sun.com/javase/ja/6/docs/ja/api/
- 基本的なクラス：java.lang
- 入出力：java.io
- コレクション(リストなど)：java.util
- 基本GUI：java.awt
- 拡張GUIセットSwing：javax.swing



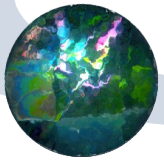
開発の手順

- 作業ディレクトリを決める
  - 例へば~/javasrc
- NetBeansを起動
- 「ファイル」->「新規プロジェクト」
- プロジェクトウィンドウ内で
  - プロジェクト名->「ソースパッケージ」->「デフォルトパッケージ」で右ボタン「新規」



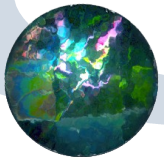
クラスを作る

- GUIの無い主クラス
  - 「Java 主クラス」
- GUIのある主クラス
  - 「Jframeフォーム」
- テンプレートを上手に使う



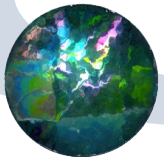
構築と実行

- プロジェクトウィンドウ内で
  - プロジェクト名->「プロジェクトを構築」
- プロジェクトウィンドウ内で
  - プロジェクト名->「プロジェクトを実行」
  - 主クラス名->「ファイルを実行」



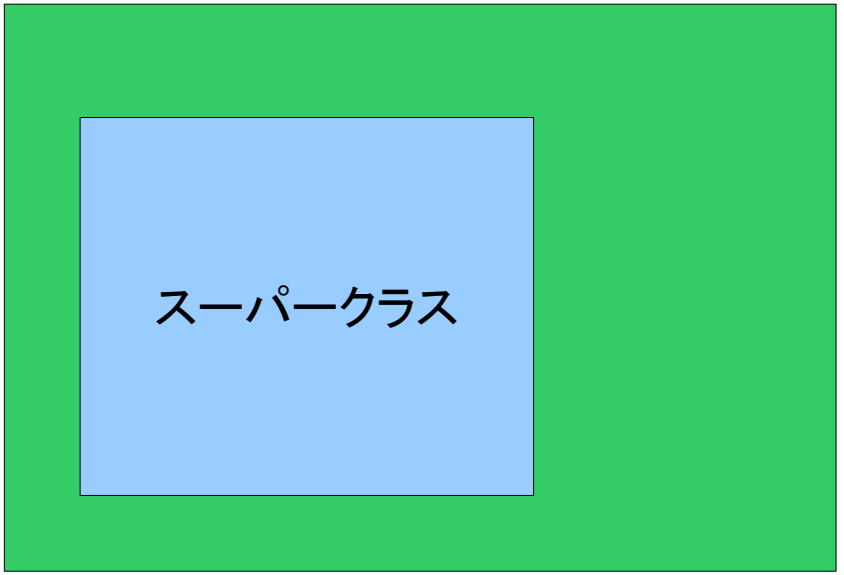
クラスの再利用

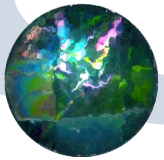
- 既存のクラスを継承して拡張
  - クラスの継承とインターフェイスの利用
- 既存のクラスとの調整をするクラスを作る
  - インターフェイス的な調整
- 既存のクラスを要素として持つクラスを作る



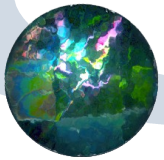
クラスの継承

- 出来上がったクラスの資産を生かす
- 標準的クラスの資産を生かす
- クラスの組に共通なデータや動作を定義する
- 一つのクラスしか継承できないことに注意



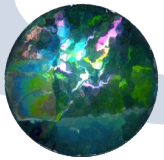


```
public class A {
 private int a; //誰もアクセスできない
 protected int b; //B はアクセスできる
 public int c; //誰でもアクセスできる
...}
public class B extends A{
...
}
```



Abstract classes

- 基本となるデータ構造とメソッドを定義
- メソッドの一部は実装が定義されていない
  - abstract method
- 継承クラスを定義して使う
- 例
  - java.util.AbstractList
  - 上記の実装の一つがjava.util.Vector



Interfaces

- abstract methodのみで構成されている
- アクセス方法だけが指定されている

class A implements インターフェイス{

}

- 例
  - java.lang.Runnable
  - メソッドrun()が定義されている
  - スレッドからの呼び出しに使う