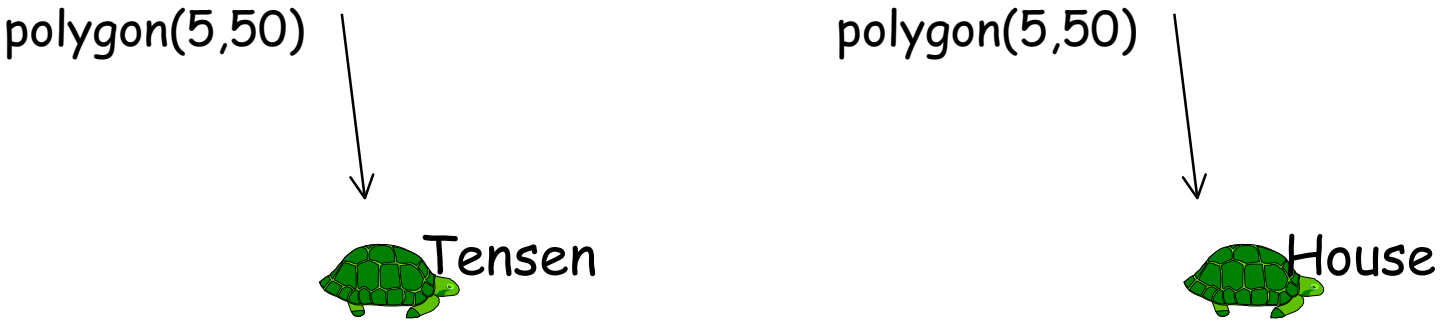
計算機プログラミングI

第10回　2002年12月19日(木)

- メソッドの再定義と動的結合
  - ◊ メソッドの再定義 (オーバーライド)
  - ◊ 型について
  - ◊ 参照型のキャスト
  - ◊ final, abstract などの修飾子
  - ◊ フィールドの隠蔽
- クイズ

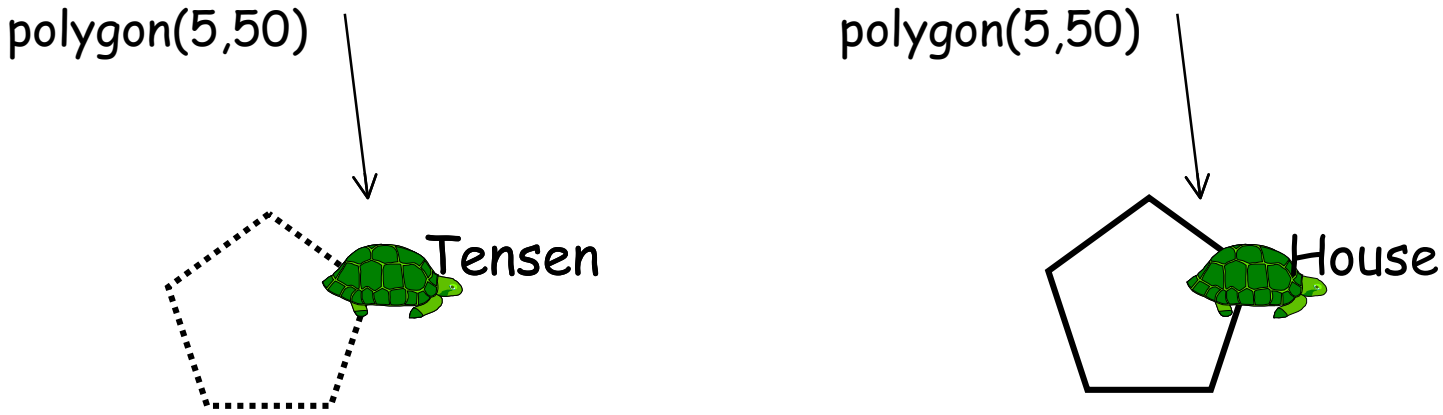
7.3 メソッドの再定義 (オーバーライド)

- 再定義とは: 親クラスで定義されているメソッドと
  同じ名前のメソッドを子クラスに定義すること
- 目的: 同じ指示に対して違った動きをさせる
- 定義の方法: 単に定義するだけ
- 例:



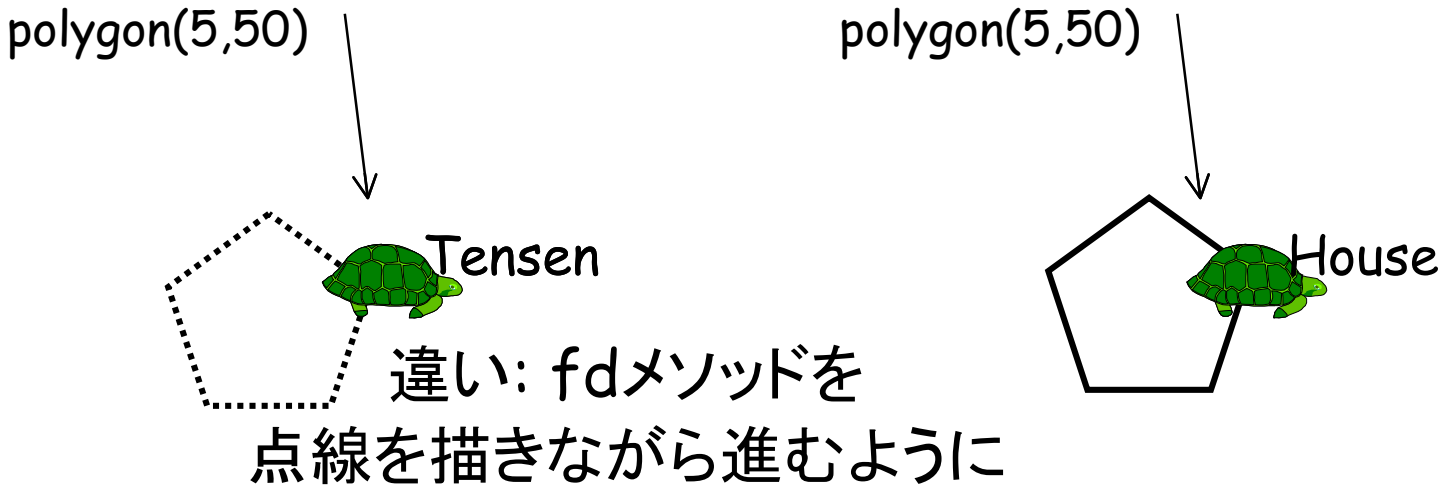
7.3 メソッドの再定義 (オーバーライド)

- 再定義とは: 親クラスで定義されているメソッドと
  同じ名前のメソッドを子クラスに定義すること
- 目的: 同じ指示に対して違った動きをさせる
- 定義の方法: 単に定義するだけ
- 例:



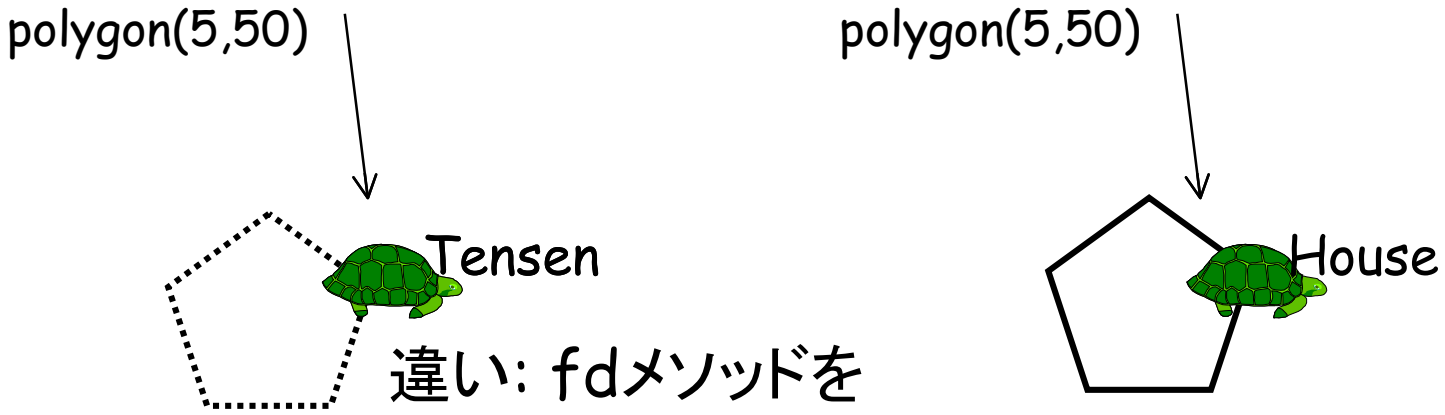
7.3 メソッドの再定義 (オーバーライド)

- 再定義とは: 親クラスで定義されているメソッドと
  同じ名前のメソッドを子クラスに定義すること
- 目的: 同じ指示に対して違った動きをさせる
- 定義の方法: 単に定義するだけ
- 例:



7.3 メソッドの再定義 (オーバーライド)

- 再定義とは: 親クラスで定義されているメソッドと
  同じ名前のメソッドを子クラスに定義すること
- 目的: 同じ指示に対して違った動きをさせる
- 定義の方法: 単に定義するだけ
- 例:



違い: fdメソッドを
点線を描きながら進むように
再定義している

点線で多角形を描く

```
public class Tensen extends House{
  int psize = 4;
  (略: コンストラクタ)
  public void fd(int s){
    長さsの点線を描く
  }

  public static void main(String[] args){
    (略: TurtleFrameの生成)
    Tensen m = new Tensen();
    (略)
    m.polygon(5, 50);
  }
}
```

- main から実行が始まる
- Tensenオブジェクトが作られる
- Tensenオブジェクトのploygonメソッドを呼び出す

- ploygonメソッドはn角形を描く。その際、Tensenオブジェクトのfdは点線を描くように再定義されているので、n角形は点線で描かれる。

点線で多角形を描

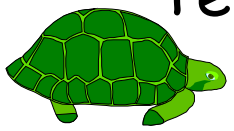
```
public class Tensen extends Turtle {
  int psize = 4;
  (略: コンストラクタ)
  public void fd(int s){
    長さsの点線を描く
  }

  public static void main(…){
    (略: TurtleFrameの生成)
    Tensen m = new Tensen();
    (略)
    m.polygon(5, 50);
  }
}
```

再定義された
fdが実行

```
public class Hou
                  extends Turtle {
  public void polygon(int n, int s){
    int a = 360/n;
    for(int j = 0; j < n; j++){
      fd(s);
      rt(a);
    }
  }
}
```

親クラスから
継承したpolygonを
実行

Tensen



点線で多角形を描

```
public class Tensen extends Turtle{
  int psize = 4;
  (略: コンストラクタ)
  public void fd(int s){
    長さsの点線を描く
  }

  public static void main(…){
    (略: TurtleFrameの生成)
    Tensen m = new Tensen();
    (略)
    m.polygon(5, 50);
  }
}
```

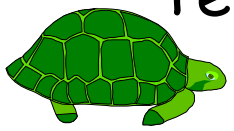
再定義された
fdが実行

ここから
スタート

```
public class Hou
                  extends Turtle {
  public void polygon(int n, int s){
    int a = 360/n;
    for(int j = 0; j < n; j++){
      fd(s);
      rt(a);
    }
  }
}
```

親クラスから
継承したpolygonを
実行

Tensen



点線で多角形を描

```
public class Tensen extends Turtle {
  int psize = 4;
  (略: コンストラクタ)
  public void fd(int s){
    長さsの点線を描く
  }

  public static void main(…){
    (略: TurtleFrameの生成)
    Tensen m = new Tensen();
    (略)
    m.polygon(5, 50);
  }
}
```

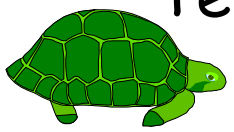
再定義された
fdが実行

オブジェクト
を作る

```
public class Hou
                  extends Turtle {
  public void polygon(int n, int s){
    int a = 360/n;
    for(int j = 0; j < n; j++){
      fd(s);
      rt(a);
    }
```

親クラスから
継承したpolygonを
実行

Tensen



点線で多角形を描

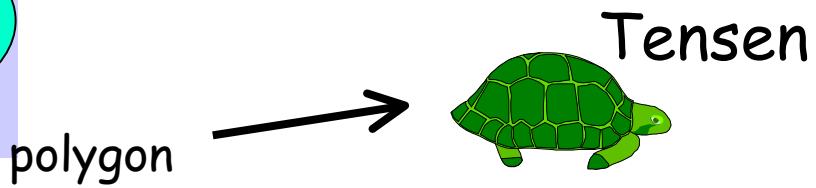
```
public class Tensen extends Turtle {
  int psize = 4;
  (略: コンストラクタ)
  public void fd(int s){
    長さsの点線を描く
  }

  public static void main(…){
    (略: TurtleFrameの生成)
    Tensen m = new Tensen();
    (略)
    m.polygon(5, 50);
  }
}
```

再定義された
fdが実行

メソッド
呼び出し

```
public class Hou
                  extends Turtle {
  public void polygon(int n, int s){
    int a = 360/n;
    for(int j = 0; j < n; j++){
      fd(s);
      rt(a);
    }
  }
}
```

親クラスから
継承したpolygonを
実行



点線で多角形を描

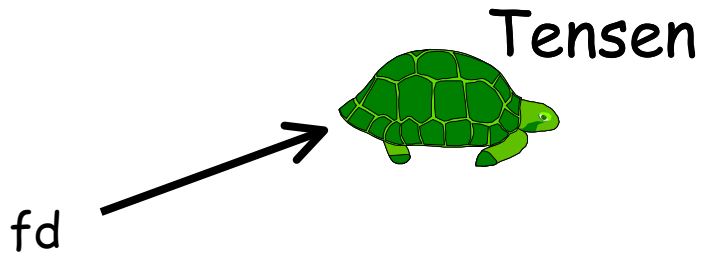
```
public class Tensen extends Turtle{
  int psize = 4;
  (略: コンストラクタ)
  public void fd(int s){
    長さsの点線を描く
  }

  public static void main(…){
    (略: TurtleFrameの生成)
    Tensen m = new Tensen();
    (略)
    m.polygon(5, 50);
  }
}
```

再定義された
fdが実行

```
public class Hou
                  extends Turtle {
  public void polygon(int n, int s){
    int a = 360/n;
    for(int j = 0; j < n; j++){
      fd(s);
      rt(a);
    }
  }
}
```

親クラスから
継承したpolygonを
実行

自身のfdを
呼び出し

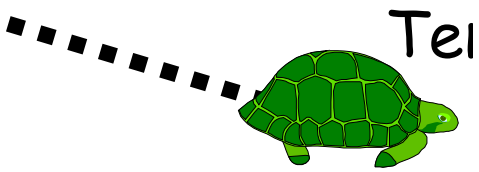


点線で多角形を描く

```
public class Tensen extends Turtle {
  int psize = 4;
  (略: コンストラクタ)
  public void fd(int s){
    長さsの点線を描く
  }

  public static void main(…){
    (略: TurtleFrameの生成)
    Tensen m = new Tensen();
    (略)
    m.polygon(5, 50);
  }
}
```

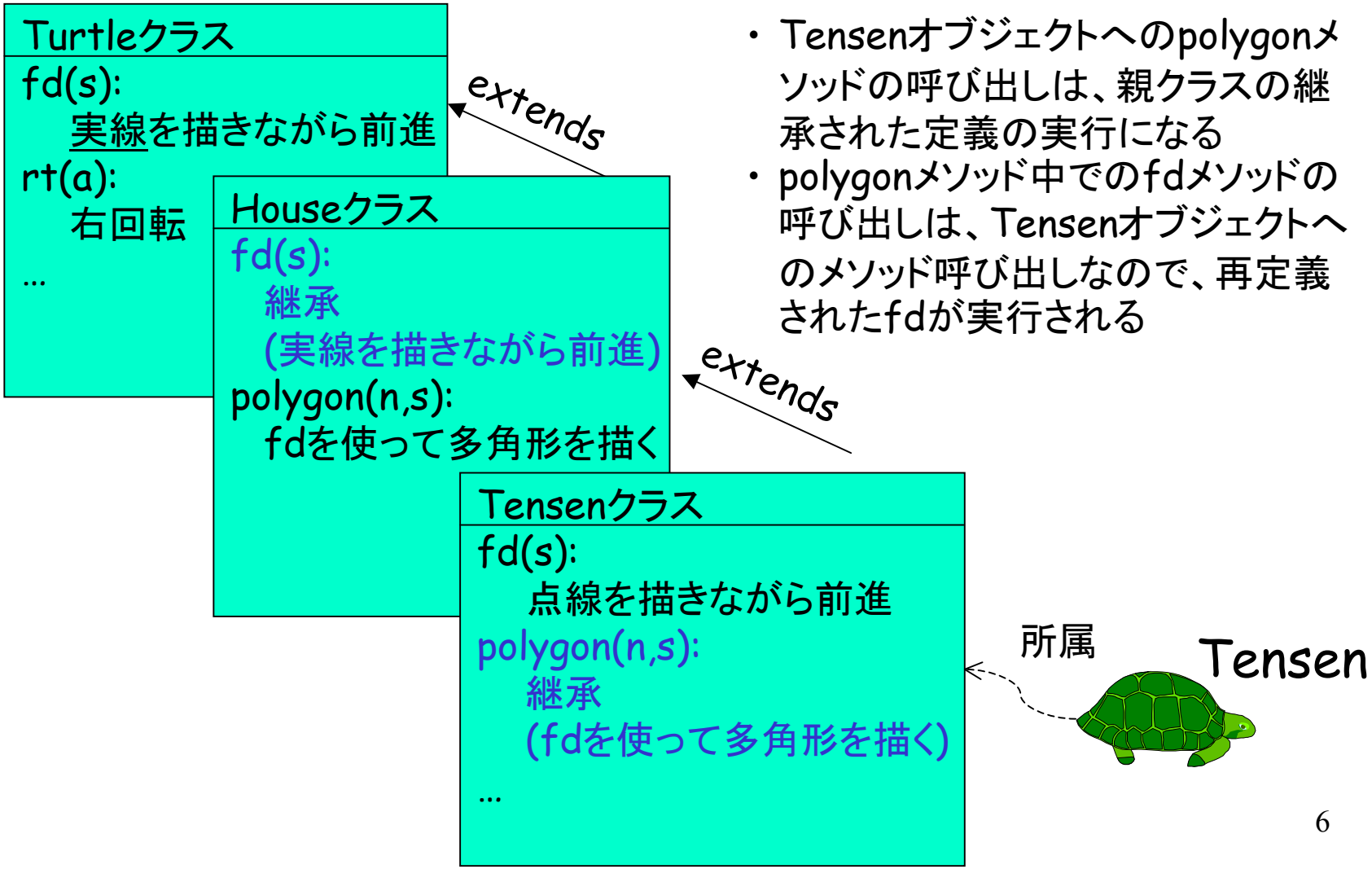
再定義された
fdが実行

```
public class House
                    extends Turtle {
  public void polygon(int n, int s){
    int a = 360/n;
    for(int j = 0; j < n; j++){
      fd(s);
      rt(a);
    }
  }
}
```



結果: 点線で5角形が描かれる

点線で多角形を描くしくみ



- Tensenオブジェクトへのpolygonメソッドの呼び出しは、親クラスの継承された定義の実行になる
- polygonメソッド中でのfdメソッドの呼び出しは、Tensenオブジェクトへのメソッド呼び出しなので、再定義されたfdが実行される

練習

(準備)* Tensen.java をコンパイルして実行し、点線で五角形が描かれることを確認せよ。

7.2* T71.javaをそのままコンパイルして実行した場合と、T71.javaを変更してHouseオブジェクトのかわりにTensenオブジェクトを作るようにした場合を試す (実行は java T71 house のようにクラス名(T71)の後に空白をあけて適当な文字列を書く。)

(教科書外) Morphing課題で作成したプログラムやフラクタル図形の「木」を描くプログラム中のTurtleオブジェクトをTensenオブジェクトに置き換えてみよ。

親クラスのメソッドを使う方法

- メソッドを再定義すると、
  親クラスのメソッド定義が使えなくなる

  例: Tensenオブジェクトのfdメソッド
- メソッド中で

  super . メソッド名(引数, ...)

  のようにメソッドを呼び出すと、
  親クラスのメソッドを実行できる

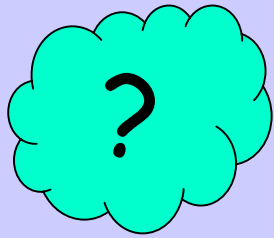
  例: Tensenオブジェクトのfdメソッド中で

  super.fd(psize)

  とすると、Turtleクラスに定義された
  本来のfdメソッドが実行される

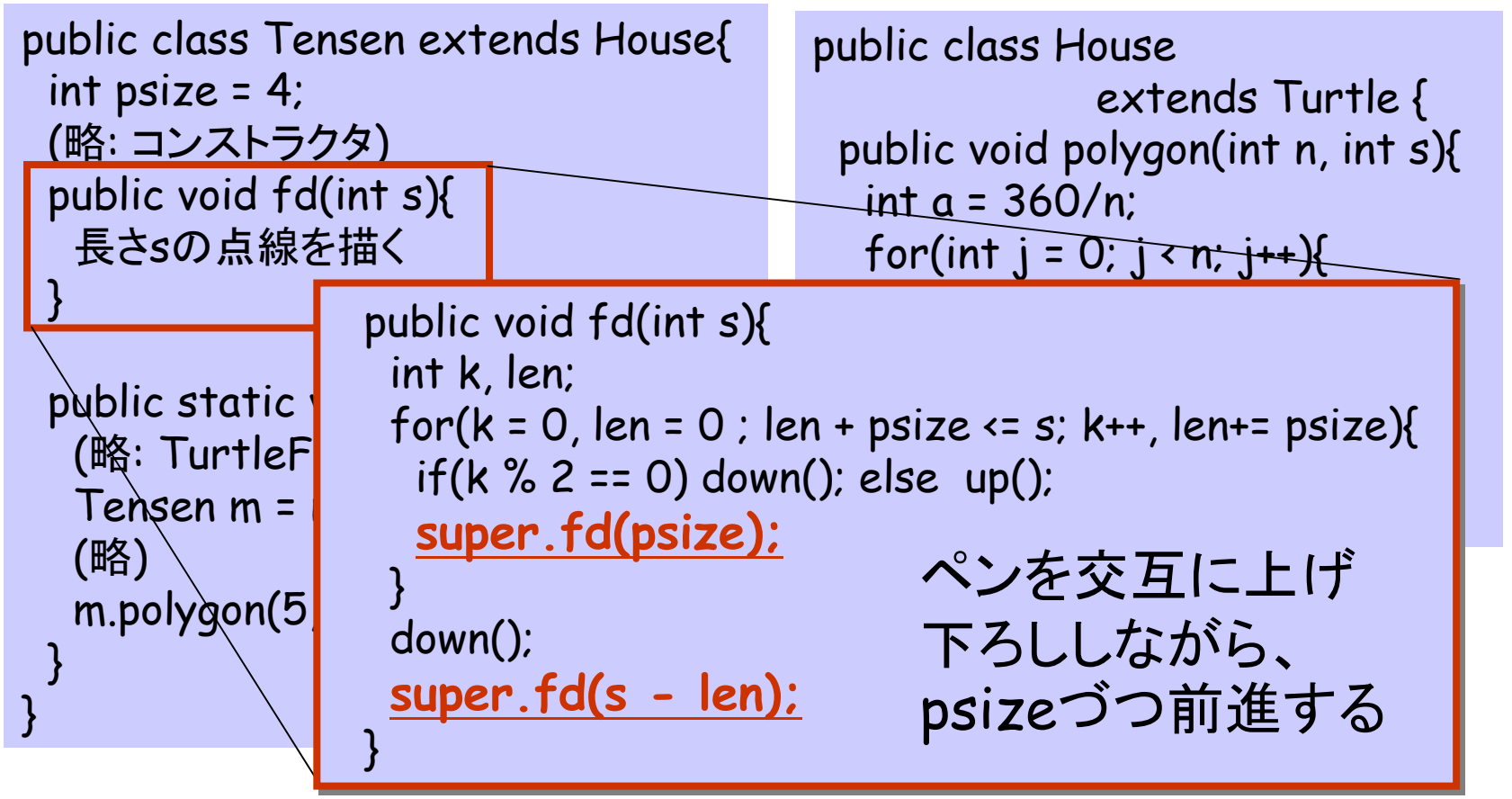
点線を描くしくみ

```
public class Tensen extends House{
  int psize = 4;
  (略: コンストラクタ)
  public void fd(int s){
    長さsの点線を描く
  }

  public static void main(…){
    (略: TurtleFrameの生成)
    Tensen m = new Tensen();
    (略)
    m.polygon(5, 50);
  }
}
```



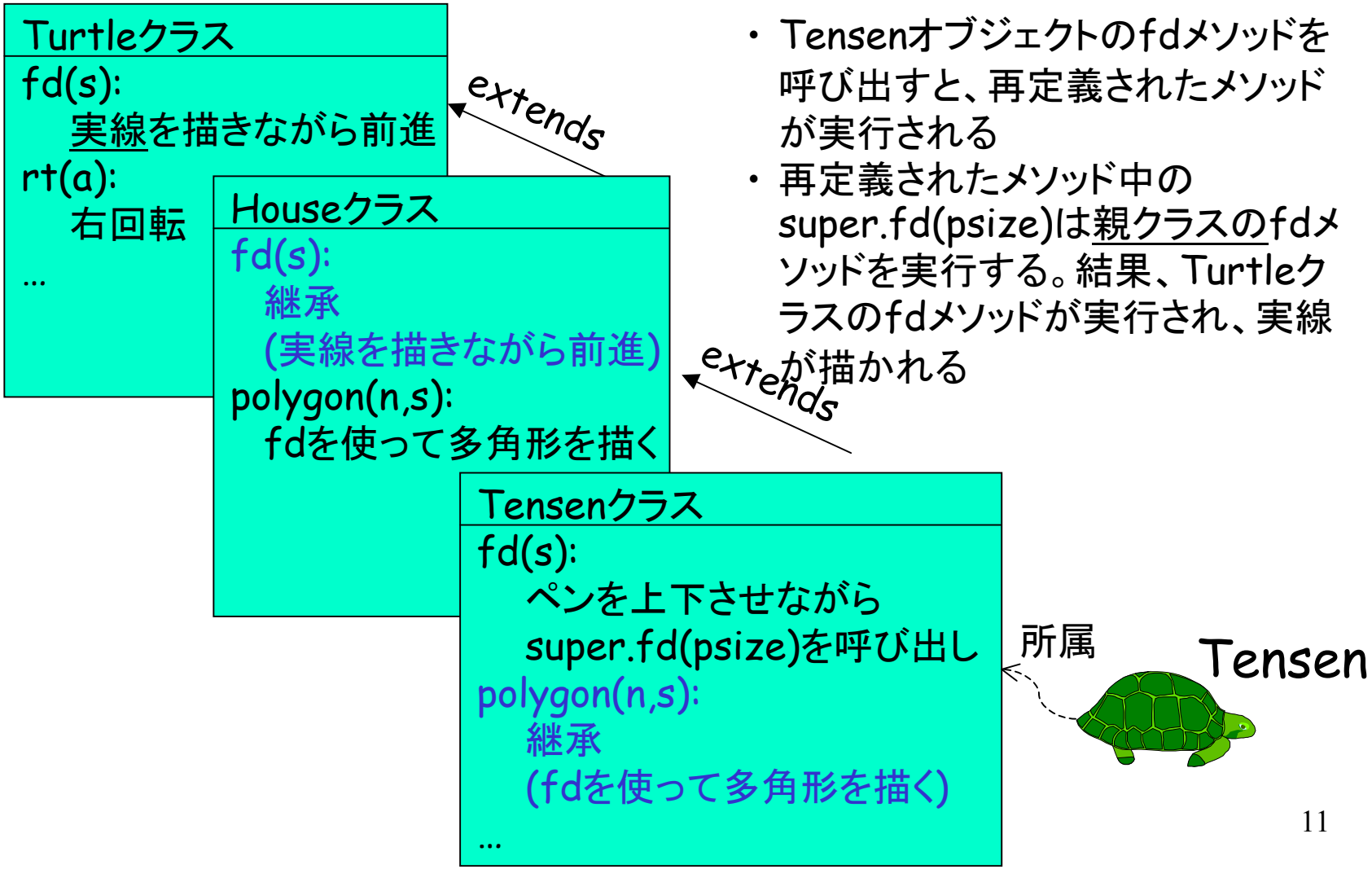

```
public class House
                    extends Turtle {
  public void polygon(int n, int s){
    int a = 360/n;
    for(int j = 0; j < n; j++){
      fd(s);
      rt(a);
    }
  }
}
```

点線を描くしくみ



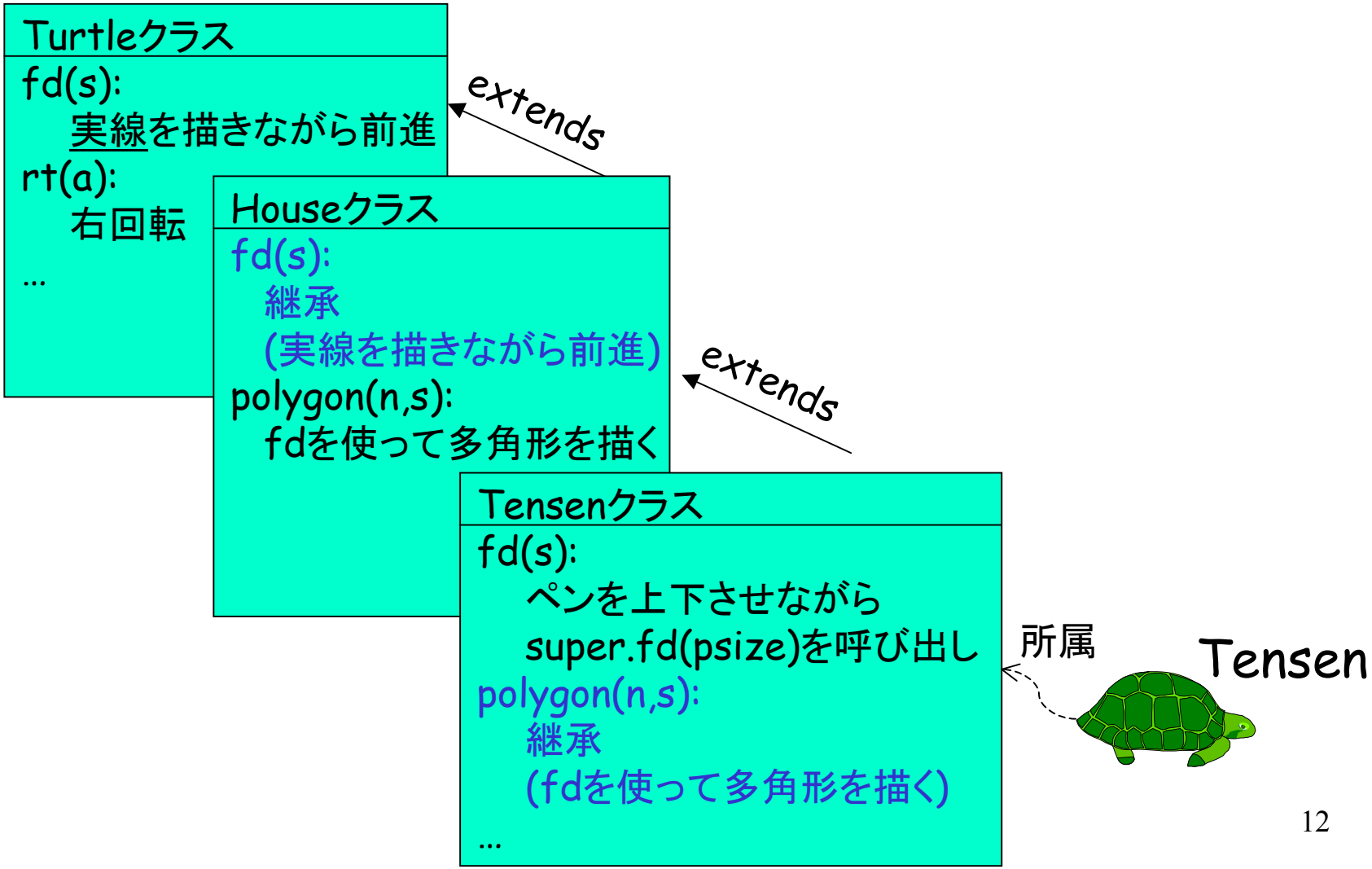
super . メソッド名(引数式, ...) 親クラスのメソッドを呼び出す

点線を描くしくみ

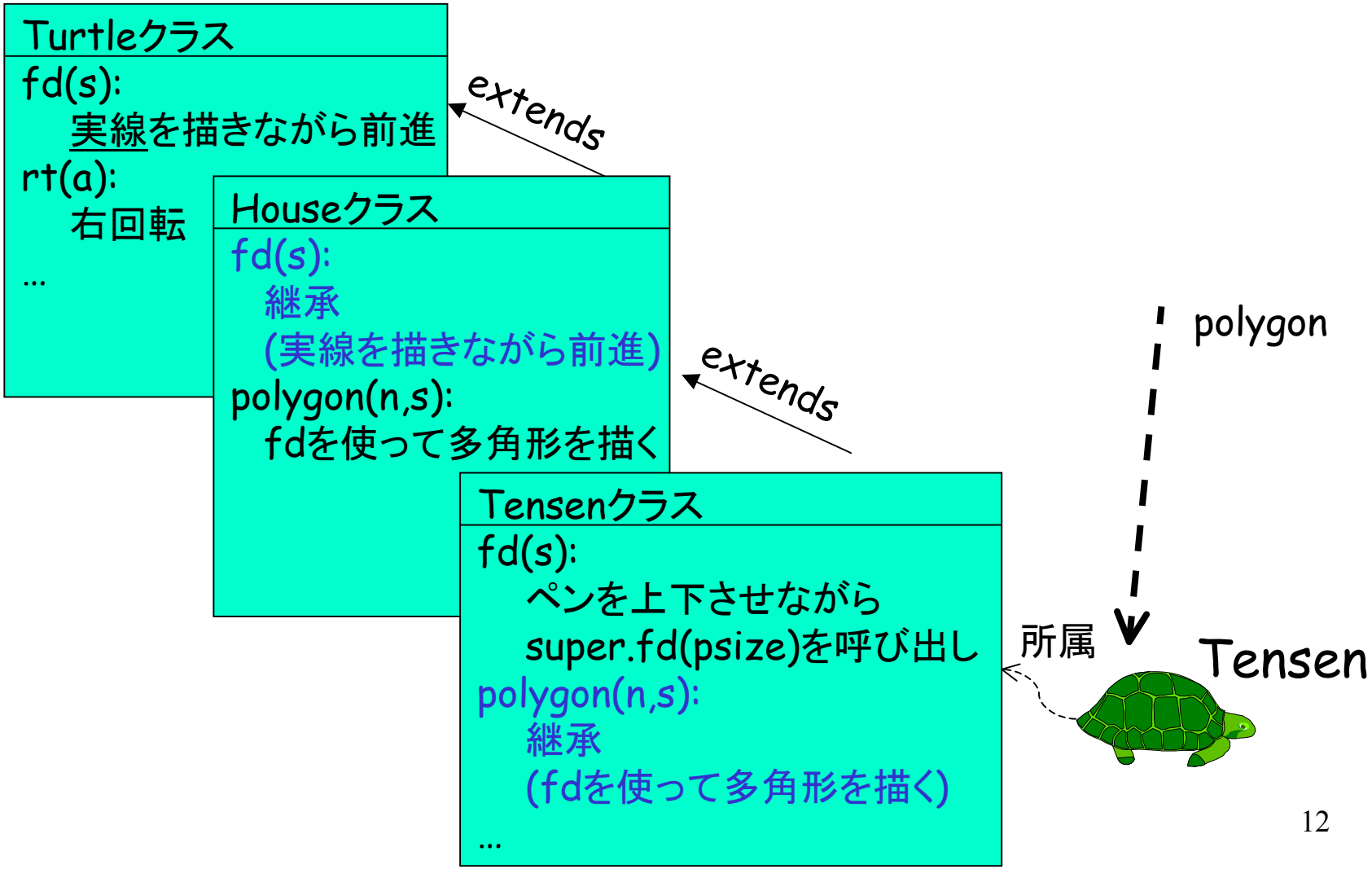


- Tensenオブジェクトのfdメソッドを呼び出すと、再定義されたメソッドが実行される
- 再定義されたメソッド中のsuper.fd(psize)は親クラスのfdメソッドを実行する。結果、Turtleクラスのfdメソッドが実行され、実線が描かれる

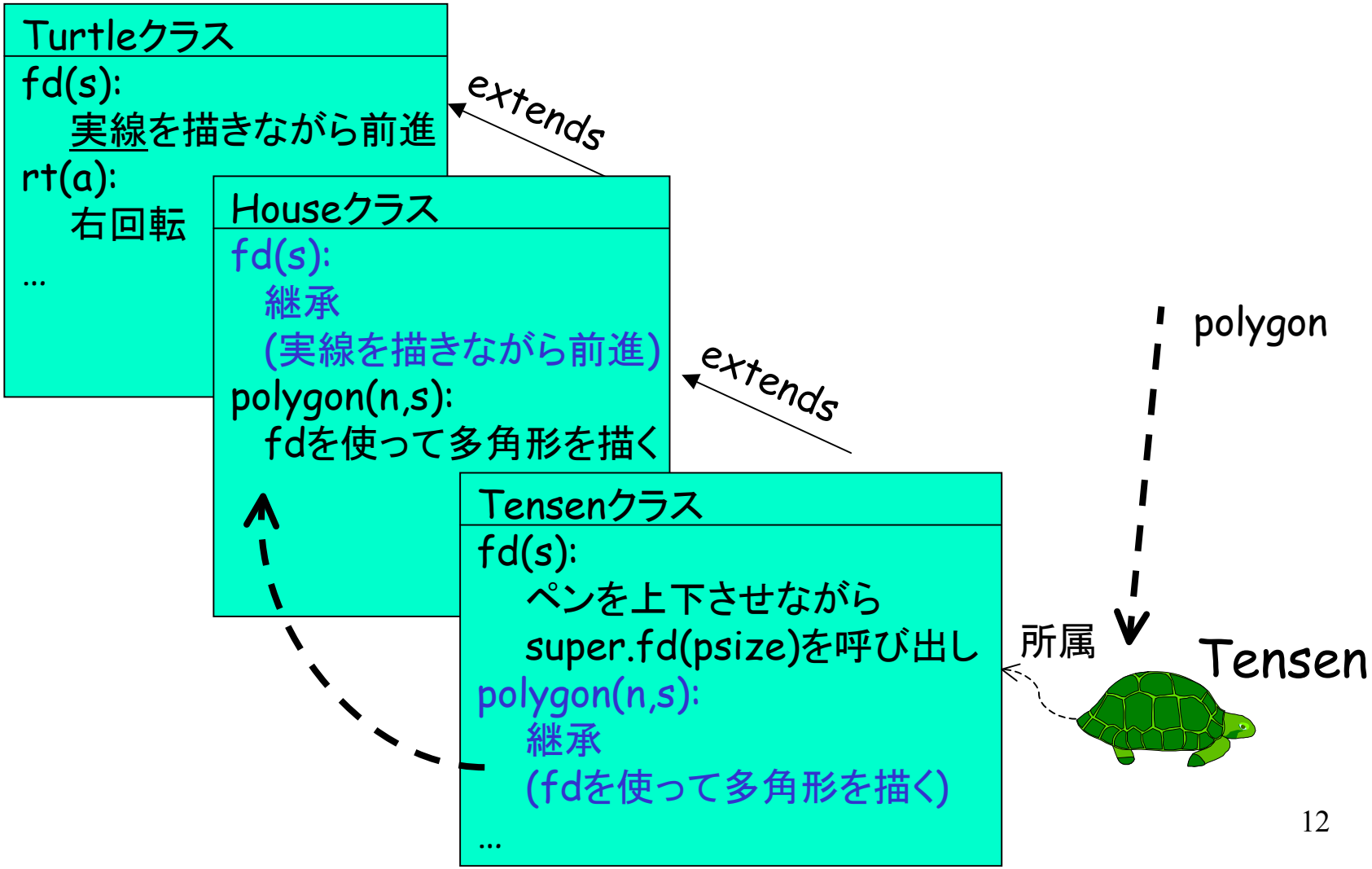
まとめると・・・


Turtleクラス
fd(s):
実線を描きながら前進
rt(a):
右回転
…
extends
Houseクラス
fd(s):
継承
(実線を描きながら前進)
polygon(n,s):
fdを使って多角形を描く
extends
Tensenクラス
fd(s):
ペンを上下させながら
super.fd(psize)を呼び出し
polygon(n,s):
継承
(fdを使って多角形を描く)
…
所属
Tensen

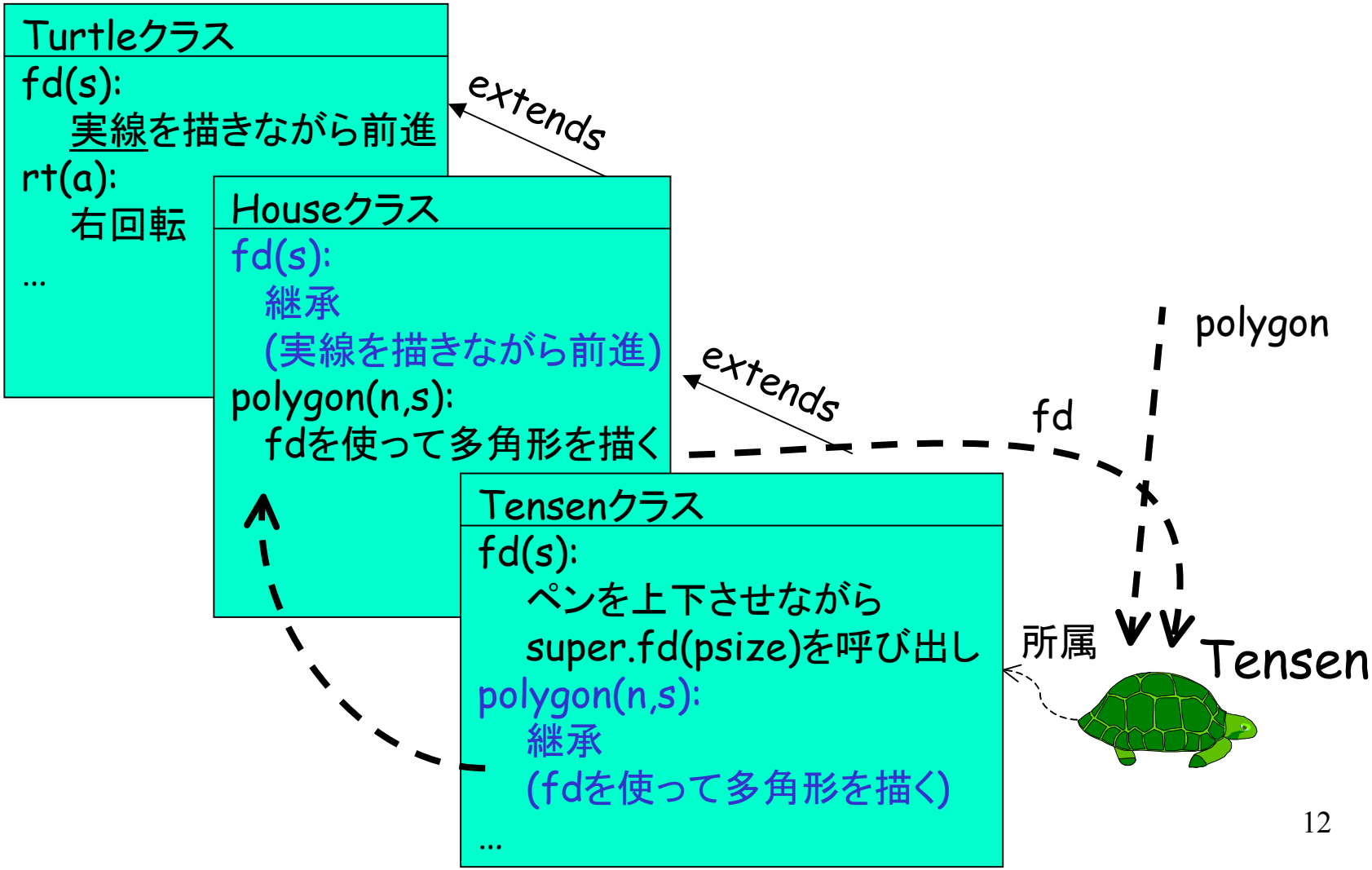
まとめると・・・


Turtleクラス
fd(s):
実線を描きながら前進
rt(a):
右回転
…
extends
Houseクラス
fd(s):
継承
(実線を描きながら前進)
polygon(n,s):
fdを使って多角形を描く
extends
Tensenクラス
fd(s):
ペンを上下させながら
super.fd(psize)を呼び出し
polygon(n,s):
継承
(fdを使って多角形を描く)
…
polygon
所属
Tensen

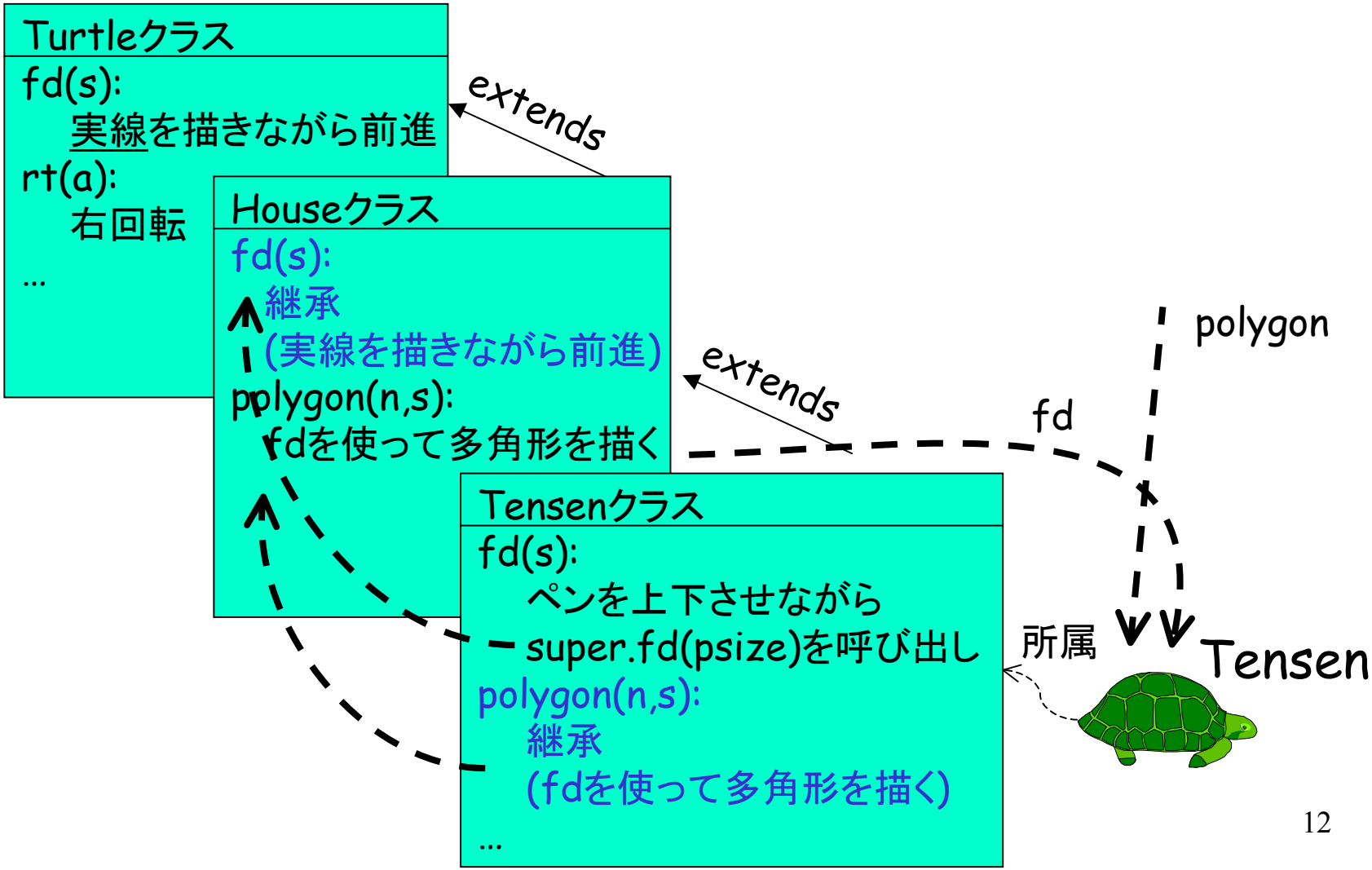
まとめると・・・


Turtleクラス
fd(s):
実線を描きながら前進
rt(a):
右回転
…
extends
Houseクラス
fd(s):
継承
(実線を描きながら前進)
polygon(n,s):
fdを使って多角形を描く
extends
Tensenクラス
fd(s):
ペンを上下させながら
super.fd(psize)を呼び出し
polygon(n,s):
継承
(fdを使って多角形を描く)
…
polygon
所属
Tensen

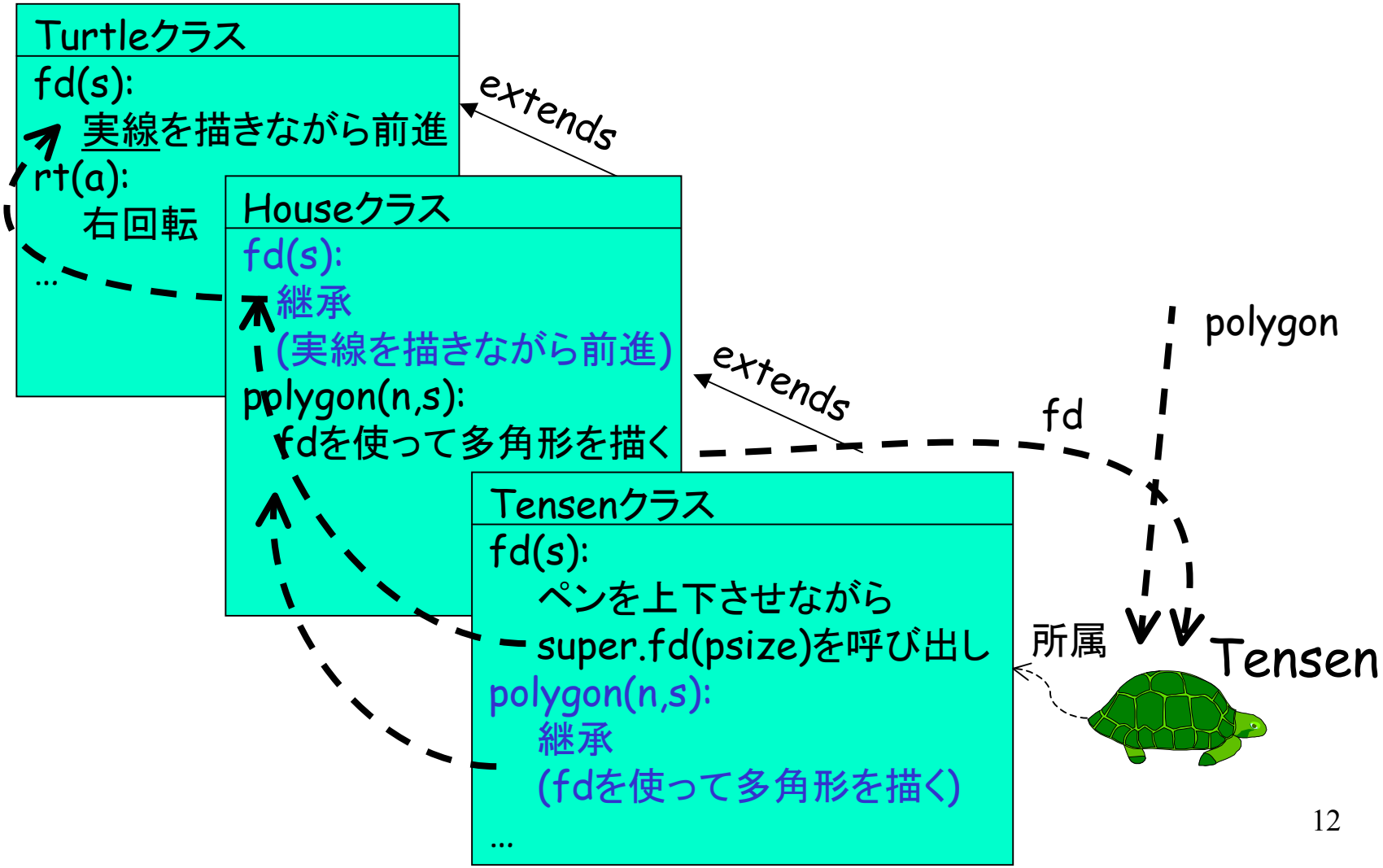
まとめると・・・


Turtleクラス
fd(s):
実線を描きながら前進
rt(a):
右回転
…
extends
Houseクラス
fd(s):
継承
(実線を描きながら前進)
polygon(n,s):
fdを使って多角形を描く
extends
Tensenクラス
fd(s):
ペンを上下させながら
super.fd(psize)を呼び出し
polygon(n,s):
継承
(fdを使って多角形を描く)
…
polygon
fd
所属
Tensen

まとめると・・・


Turtleクラス
fd(s):
実線を描きながら前進
rt(a):
右回転
…
extends
Houseクラス
fd(s):
継承
(実線を描きながら前進)
polygon(n,s):
fdを使って多角形を描く
extends
Tensenクラス
fd(s):
ペンを上下させながら
super.fd(psize)を呼び出し
polygon(n,s):
継承
(fdを使って多角形を描く)
…
polygon
fd
所属
Tensen

まとめると・・・


Turtleクラス
fd(s):
実線を描きながら前進
rt(a):
右回転
...
extends
Houseクラス
fd(s):
継承
(実線を描きながら前進)
polygon(n,s):
fdを使って多角形を描く
extends
Tensenクラス
fd(s):
ペンを上下させながら
super.fd(psize)を呼び出し
polygon(n,s):
継承
(fdを使って多角形を描く)
...
polygon
fd
所属
Tensen

練習

7.3 注16のヒント参照。
ペンが下りてなかったら、
親クラスのfdと同じ動作をすればよい

7.4

7.5

7.1 型について

- 型の目的: 処理と値の不一致を防ぐ
  - ◊処理と値の不一致の例:
    - 123 / “hello“
    - 123 . fd(456)
    - (Turtle m = new Turtle(); のときに)
      123 * m
    - m . polygon(5,50)
  - ◊「型」を考えると、コンパイルしたときに
    問題があることが分かる

7.1 型について

- 型とは: 変数・引数・返値などがとり得る値の集合
  - ◊例: int, double, Turtle, Complex, ...
- 型を使った検査:
  - ◊式 → その式から得られる値の集合が型
  - ◊処理(演算・メソッド呼出等)→関係する式の型を調べる
    - 123 / "hello" → int / String
    - 123 . fd(456) → int . fd(int)
    - (Turtle m = new Turtle(); のときに)
      123 * m → int * Turtle
    - m . multiply(m) → Turtle.multiply(Turtle)

プログラムを実行しなくても問題が分かる

オブジェクトと型

- m.fd(10) という式は正しいか?
- m にしまわれるオブジェクトのクラスに、
  いつもfdというメソッドが定義されていれば正しい
  (クラスはいつも同じとは限らない)
- Java言語では:
  - ◊変数mに型をつける (型=クラス)　　Turtle m;
  - ◊そのクラスにfdメソッドが定義されていれば正しい
  - ◊mにはそのクラスのオブジェクトしか代入できない

を許してもこれは守られる

オブジェクトと型

- m.fd(10) という式は正しいか?
- m にしまわれるオブジェクトのクラスに、
  いつもfdというメソッドが定義されていれば正しい
  (クラスはいつも同じとは限らない)
- Java言語では:
  - ◊変数mに型をつける (型=クラス)　　Turtle m;
  - ◊そのクラスにfdメソッドが定義されていれば正しい
  - ◊mにはそのクラスのオブジェクトしか代入できない

とサブクラス

を許してもこれは守られる

7.2 参照型とキャスト

7.4 final, abstractなどの修飾子

7.5 フィールドの隠蔽

(略)

クイズ