H7 春 PE 午後 1

問2　オブジェクト指向設計に関する次の記述を読んで,設問 1 ～ 3 に答えよ.

図に示すオブジェクト指向モデルのダイアグラムは，2 次元の幾何学図形についてのクラスを示している。これは, J.Rumbaugh 等による OMT 表記法に従っている。

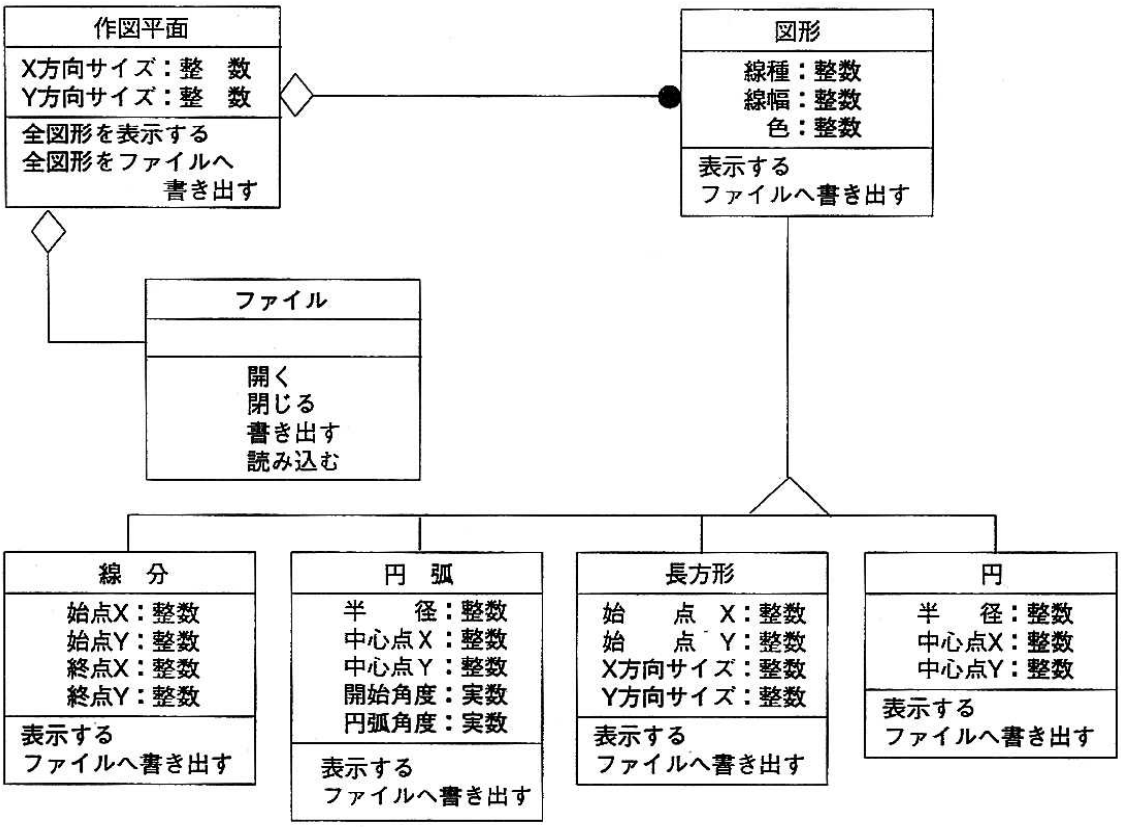
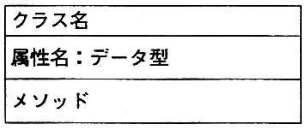


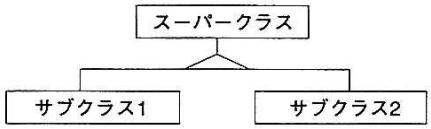
図　２次元幾何学図形のＯＭＴ表記ダイアグラム

OMT 表記法のダイアグラムでは,次のような図式表現が使われる。

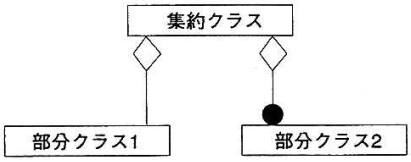
(a)クラスの詳細表記



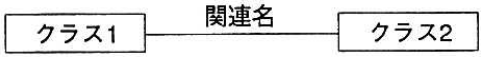
（b）継承関係（is -a の関係）



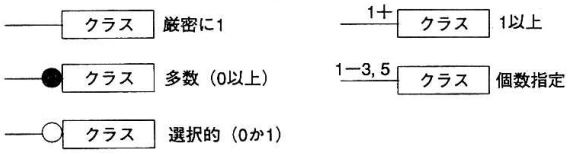
(c)集約関係(has-a 関係)



(d)関連



(e)関連及び集約の多重度



図の各クラスにおけるメソッドと属性は,次のとおりである。

(1) 作図平面クラスのメソッド "全図形をファイルへ書き出す" の仕様

(a) 所定のファイルを開く。

(b) 図形クラスの各インスタンスのメソッド "ファイルへ書き出す" を呼び出す。

(c) 所定のファイルを閉じる。

(2) 図形クラスの各サブクラスのメソッド "ファイルへ書き出す" の仕様

(a) 図形クラスの各サブクラスのクラス名を書き出す。

(b) 各属性の値を変数に読み込む。

(c) (b)の変数の値をファイルへ書き出す。

(3) 図形クラスの属性とメソッドは,サブクラスである線分,円弧,長方形,円の各クラスに継承されている。ただし, "表示する", "ファイルへ書き出す"のメソッドにはポリモルフィズム(多相性)が用いられている。これらのメソッドが呼び出されると,サブクラスに用意された "表示する", "ファイルへ書き出す" のメソッドが使用される。

(4) 各クラスの属性は私用(private)メンバであり,メソッドは公開(public)メンバであるとする。そのため,各属性へのアクセスは,すべてメソッドを介さなければならない。属性にアクセスするメソッドには, "読み込む", "書き出す" の 2 種類があり,必ず 1 属性に対し "読み込む", "書き出す" の二つのメソッドを用意する。

設問1　図には,各属性へアクセスするためのメソッドが明記されていない。クラスに "読み込む", "書き出す" の 2 種類のメソッドを追加して,各属性へのアクセスが,必ずこの 2 種類のメソッドを介して行われるようにするためには,何個のメソッドを設計しなければならないか。メソッドの個数を求めよ.

設問2　図に示された設計にしたがってプログラムを作成し,ある作図を実行したところ,次表に示される個数の図形クラスの各サブクラスのインスタンスが生成され,作図平面上に表示された。

クラス	インスタンス個数
線　分	2
円　弧	3
長方形	1
円	2

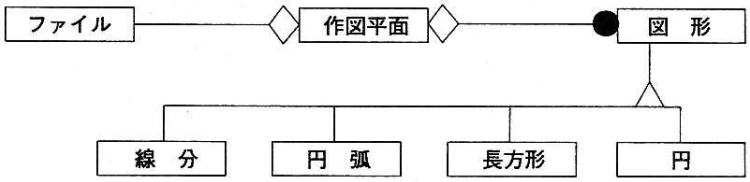
このとき,作図平面クラスの "全図形をファイルへ書き出す"メソッドによつて,作図平面上に表示されている図形クラスの各サブクラスのインスタンスのデータを順次書き出したい。ファイルに書き出される総バイト数を求めよ。ただし,条件は次の

とおりである。

(a) 各属性の型はすべて基本データ型とし,整数型と実数型の 2 種類を用いている。整数型は 2 バイト,実数型は 4 バイトのデータ型である。

(b) クラス名は 8 バイト固定長であり,図形クラスの各サブクラスの "ファイルへ書き出す"メソッドのプログラム内に埋め込まれている。

設問3 (1)図形クラスのメソッドについて,何らかの理由で,ポリモルフイズムが使用できない場合,図をどのように手直しするのがよいか。図式表現を用いて,次のダイアグラム上に追加するクラス間の関係を図示せよ。ここでは,スーパークラスのインスタンスをサブクラスに型変換できないものとする。



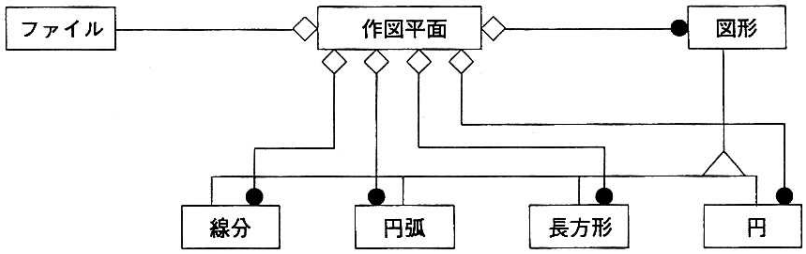
(2) (1)の場合に,図形クラスを継承した新たなサブクラスを追加するとき,設計内容を変更しなければならないクラス名と,その理由を 50 字以内で述べよ。

《解答》

[設問 1]　42(個)

[設問 2]　19)(バイト)

「設問 31 (1)



(2)（クラス名)作図平面クラス

(理由）サブクラスを追加するごとに,作図平面クラスでサブクラスへ振り分けする機能を追加する必要がある。

《解説》

オブジェクト指向は,データを手続きによって処理するのではなく,データとその手続きをオブジェクトという単位にまとめ,オブジェクト間でメッセージのやりとりをして処理を進めるという考え方である。たとえば,図形(長方形や円など)を描くことを考えてみよう。手続き型処理では,手続き部が図形の種類を判断して図形を描くのに対して,オブジェクト指向型では,各オブジェクトに　"図形を描け"　とメツセージを送ることで,各オブジェクトが自立的に判断して図形を描く。

[設問 1]

"読み込む", "書き出す" の 2 種類のメソッドを介して各属性へアクセスしなければいけないということは,属性一つに対して二つのメソッドが必要ということである。属性は, OMT 表記法の(a)に示されているとおりである。属性は,次のように全部で 21 だから,追加すべきメソッドの個数は次のようになる。

メソッドの個数 = 21 × 2 = 42

クラス	属性の個数
作図平面クラス	2
図形クラス	3
ファイル	0
線分	4
円弧	5
長方形	4
円	3
計	21

[設問 2]

インスタンス(実体)は,あるクラスの定義をひな型として,実際に作られたオブジェクトである。インスタンスはあくまで一つの例であるため,同じクラスに属する複数のインスタンスはそれぞれ異なった値(データ)をもつ。たとえば,長方形は四つの辺を持つという性質は同じであるが,辺の長さはそれぞれ違う。だから,描こうとする長方形の数だけインスタンスが生成される。例えば,線分インスタンス 1 個についてみると,整数型の属性が四つあり,整数型は 2 バイト,さらにクラス名が 8 バイト固定だからファイルに書き出される容量は次のようになる。

線分インスタンス 1 個のバイト数 = 2 × 4 + 8 = 16(バイト)

さらに,インヘリタンス(継承)を考える。線分クラスは,スーパクラスの図形の属性を引き継ぐ。図形クラスには,三つの整数型の属性があるため,(2 × 3 =)6 バイトが線分インスタンスに含まれる。したがって,線分インスタンス一つの容量は次のようになる。

線分インスタンス 1 個の容量 = ファイル名 + 継承した属性 + 線分クラスの属性

= 8 + 6 + 8 = 22(バイト)

線分インスタンスの数が 2 個だから,線分クラスの全容量は次のようになる。

線分クラスの全容量 = 22 × 2 = 44(バイト)

他のクラスについても同様に考えると,次表のようになる。

表　ファイルに書き出される全容量

クラス	名前	継承	クラス属性	インスタンス容量	生成数	全容量
線分	8	6	整数4 4×2＝8	8＋6＋8＝22	2	22×2＝44
円弧	8	6	整数3、実数2 3×2＋2×4＝14	8＋6＋14＝28	3	28×3＝84
長方形	8	6	整数4 4×2＝8	8＋6＋8＝22	1	22×1＝22
円	8	6	整数3 3×2＝6	8＋6＋6＝20	2	20×2＝40
					合計	190

［設問 3］

(1) ポリモルフイズムが使えないと,問題文中の(3)の説明にある "図形クラスのメソッドが呼び出されると,サブクラスに用意されたメソッドが使用される" ができなくなる。すなわち,作図平面クラスから図形クラスの "ファイルへ書き出す"メソッドが呼ばれても,サブクラスのメソッドは呼ばれないため,線分や円弧,長方形,円の各図形がファイルへ書き出されない。そこで,作図平面クラスのメソッドから線分,円弧,長方形,円のクラスのメソッドを直接呼び出せるように,作図平面クラスと線分,円弧,長方形,円のクラスの間に集約関係を結ぶ必要がある。数の対応は,各図形とも作成数の制限がないので, "0 以上" となる。

(2) ポリモルフイズムが使えるならば,図形クラスが必要に応じて各サブクラスへと処理を振り分け,各サブクラスのメソッドを呼び出すことができる。しかし, (1)の状態であると,各サブクラスのメソッドを作図平面クラスが呼び出すとき,それが線分なのか円弧なのか長方形なのか円なのかを認識することができない。したがって,作図平面クラスのメソッドに,線分,円弧,長方形,円の各サブクラスへ振り分ける機能を追加する必要がある。