LastUpdate : 2009/11/12

特定非営利活動法人 UML モデリング推進協議会

UML モデリング技能認定試験

L3　サンプル問題解答

モデリング問題-解答 4（ビジネス系）

解答

設問 1

正解

a. （1）
b. （4）
c. （2）
d. （5）
e. （6）
f. （3）

解説

解答欄に当てはまらない選択肢がないため、クラス候補を選定する作業は省略できます。関連クラスや、合成集約、多重度や属性などから答えを導いていきます。単にクラスをひとつずつ解答欄に割り当てていくのではなく、いくつかの条件を考慮して、総合的に判断する必要があることに注意してください。必要であれば、設問 2 の属性の検討と同時に行うとよいでしょう。この流れは、実際のソフトウェア開発の中でも同じように発生します。暫定的に抽出したクラスに対する属性や操作を検討する中で、クラス構造の見直しも行うのです。

解答の手順は、いくつか考えられますが、ここでは、「フットサルコート運営会社」、「試合」を足がかりとした方法を解説します。この場合の、クラスの発見順序は次の通りです。

「フットサルコート運営会社」（既出）⇒
「試合」⇒「チーム」⇒「試合結果」⇒「大会参加」⇒「大会」⇒「コート」

最初に、属性「開始時刻」と「終了時刻」を持つクラスを識別します。このクラスに該当する選択肢として、「表 2 試合結果表の説明」をもとに、「試合」と「試合結果」の2つのクラスを候補としてあげることができます。実際のソフトウェア開発では、問題記述に相当する資料や、ヒアリング結果に登場する名詞をドメイン用語集とし、クラス名の候補とします。ここでは、一旦「試合」を仮採用して、別の観点からの検討を進めることとします。他の検討の結果、「試合」よりも「試合結果」の方がふさわしい場合は、「試合結果」を採用し直した上で再度検討を行います。

次に、「フットサルコート運営会社」に合成集約されるクラスを識別します。選択肢にある6つのクラスのオブジェクトは、フットサルコート運営会社固有のものであるため、前述の「試合」以外の全ての選択肢が候補となり得ます。

「フットサルコート運営会社」に合成集約されるクラスへの多重度には大きな違いがないため、クラスを判別する決定的な材料にはなりません。そこで、「フットサルコート運営会社」に合成集約されるクラス以外の残りのクラスが、すべて関連クラスであることに着目します。つまり、他のオブジェクト同士のリンク毎にのみ存在できるオブジェクトかどうかを判断していくのです。問題記述から、「コート」「大会」「チーム」の3つのクラスは、他のオブジェクト毎に存在するものではなく、「フットサルコート運営会社」単位に、単独で存在できるものであると想定できます。

次に、これら3つのクラス「コート」「大会」「チーム」を、解答欄（ａ）（ｂ）（ｃ）のどこに割り当てるかを決定します。これを行うには、既に明らかになっているクラスである、「試合」クラスとの関係に着目します。

「フットサルコート運営会社」に合成集約される3つのクラスと、「試合」クラスの関係には、通常の関連、更なる合成集約、関連クラスを伴う関連の3つがあります。このうち、関連クラスを伴う関連は多重度が 2 であり、最も特徴的な関係と言えます。「試合」と関連するクラスのオブジェクトの

うち、必ず2つリンクするものは、「コート」でも「大会」でもなく、「チーム」であると言えます。これは、「図1成績表の例」からも判断することができます。

次に、「試合」と「チーム」の関連に存在する関連クラスに着目します。関連クラスとなるクラスは、「試合結果」と「大会参加」の2つであることがわかっています。問題記述では、「試合結果」という名詞は、「図1　成績表の例」の中で登場します。これにより、「チーム」から見て、「試合」毎に存在するクラスは、「試合結果」が妥当であると判断できます。

「チーム」には、もう1つ、関連クラスを伴う関連があります。消去法により、この関連クラスは「大会参加」となります。この「大会参加」クラスは、設問 2 で登場する「エントリーナンバー」をはじめとした、「大会」毎の「チーム」の属性を保持します。「チーム」から見て、「大会参加」を介して関連するクラスは「大会」です。

最後に、残った「コート」を配置します。次の「図 3 設問 1 の解答」は、設問 1 を解答した後のクラス図です。

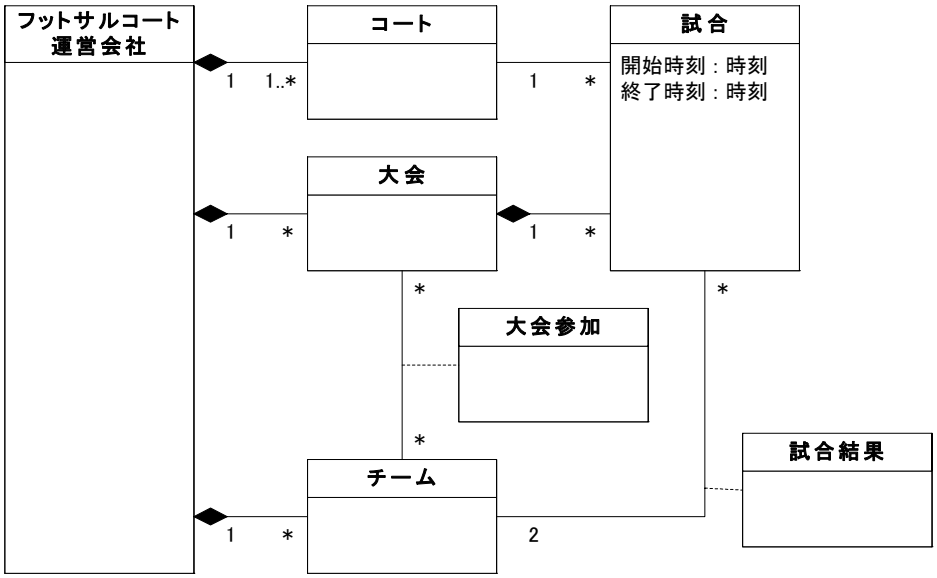

図 3 設問 1 の解答

設問 2

正解

(1) d
(2) a
(3) e
(4) f
(5) b
(6) d
(7) d
(8) f

解説

設問 1 で解答したクラスに属性を割り当てます。必要であれば、属性を検討する最中に、設問 1 に立ち返り、クラスの配置を変更します。

比較的解答しやすい属性は、「エントリーナンバー」「コート名」「チーム名」「開催日」です。それぞれ、問題記述をよく読み、どのオブジェクトの単位に存在する情報かを意識すれば、解答することができます。次に、「得点」と「反則数」は、一回の「試合」に対して、チーム毎に存在する値です。このため、「チーム」から見て、「試合」毎に存在する概念である「試合結果」に割り当てることができます。残りの項目は、主に「図1　成績表の例」にある、「勝ち点表」に印字する項目です。これらの項目は、「チーム」から見て、「大会」毎に存在するため、該当する概念は、「大会参加」になります。次の「図 4 設問 2 の解答」は、設問 2 を解答した後のクラス図です。

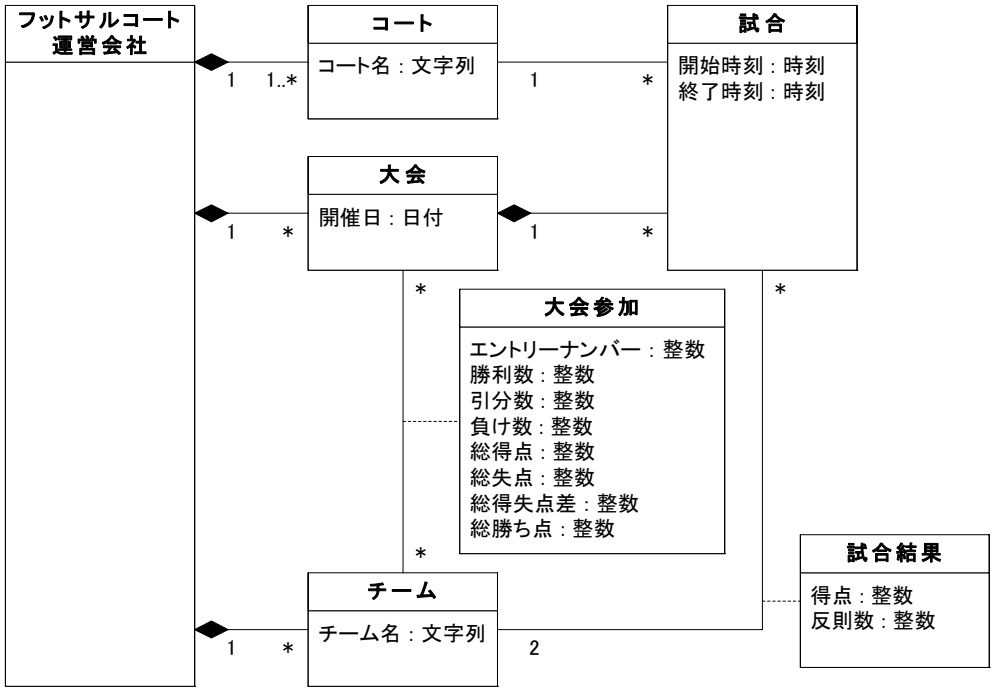

図 4 設問 2 の解答

設問 3

正解

(4)、(6)、(7)、(8)、(9)、(10)、(13)

解説

設問 2 で解答した属性のうちのいくつかを、派生属性とする問題です。派生属性とは、他の属性の値から算出するなどして、機械的に値を導き出せる属性です。派生属性を表記する際には、属性名の先頭に "/" (スラッシュ)を付けて表します。次の「図 5 派生属性の例」の「年齢」は、「誕生日」の値と、現在の日付から計算することができるため、派生属性としています。

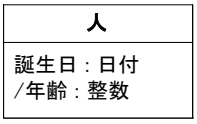

図 5 派生属性の例

なお、派生属性の値を求める際に使用できる他の属性の値は、対象の派生属性が所属するオブジェクトの属性の値だけではなく、リンクをたどって取得できる、他のオブジェクトの属性の値も使用することができます。

また、今回の問題では登場しませんが、派生属性と同様に、派生関連端や、派生関連も表すことができます。これは、他の関連や制約などから機械的に導き出せる関連です。次の「図 6 派生関連端、派生関連の例」の「会社」から「人」への関連「雇用する」と、その関連端「社員」は、「雇用」を経由した別の関連の経路をもとに導き出すことができるため、派生関連端、派生関連としています。

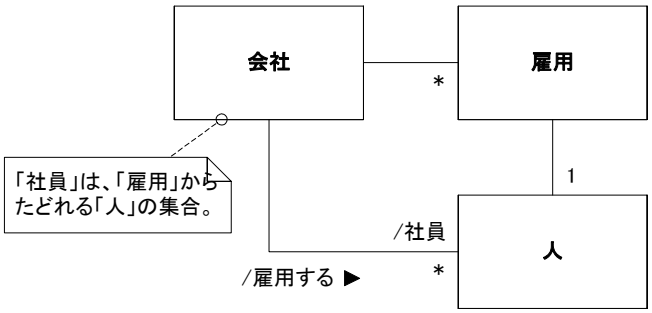

図 6 派生関連端、派生関連の例

今回の問題では、「表 1　勝ち点表の項目説明」の「勝ち点表」の項目説明から、いくつかの属性を派生属性とすることができます。派生属性とできる属性と、その導出方法の一例を、次の「表 4 派生属性と導出方法」に示します。

派生属性	導出方法例
勝利数	1. 対象の「大会参加」から、ひもづく「チーム」を参照する。 2. 手順 1 の「チーム」に関連する「試合」と「試合結果」のうち、対象の「大会参加」にひもづく「大会」に関連するものを1つ取得する。 3. 手順 2 の「試合」に関連する、もう一方の「チーム」（対戦相手）にひもづく「試合結果」を取得する。 4. 手順 2 の「試合結果」と手順 3 の（対戦相手の）「試合結果」の「得点」を比較し、手順 2 の「試合結果」の方が大きい場合は、勝利数としてカウントする。 5. 手順 2～4 を、対象となる「大会」および「チーム」に関連する、全ての「試合」に対して繰り返す。
引分数	勝利数と同じ方法で、「試合結果」の「得点」が対戦相手と同じものをカウントする。
負け数	勝利数と同じ方法で、「試合結果」の「得点」が対戦相手よりも小さいものをカウントする。
総得点	手順 1～2 は、勝利数の手順と同じ。

	3. 「試合結果」の「得点」を加算する。 4. 手順 2～4 を、対象となる「大会」および「チーム」に関連する、全ての「試合結果」に対して繰り返す。
総失点	手順 1～3 は、勝利数の手順と同じ。 4. （対戦相手の）「試合結果」の「得点」を加算する。 手順 2～4 を、対象となる「大会」および「チーム」に関連する、全ての「試合結果」に対して繰り返す。
総得失点差	手順 1～3 は、勝利数の手順と同じ。 4. 対象の「チーム」の「試合結果」の「得点」から、対戦相手の「試合結果」の「得点」を引いた値を加算する。 手順 2～4 を、対象となる「大会」および「チーム」に関連する、全ての「試合結果」に対して繰り返す。
総勝ち点	勝利数と同じ方法で、「試合結果」の「得点」が対戦相手よりも大きい場合は3点、等しい場合は1点として加算する。

表 4 派生属性と導出方法

次の「図 7 設問 3 の解答」は、設問 3 を解答した後のクラス図です。

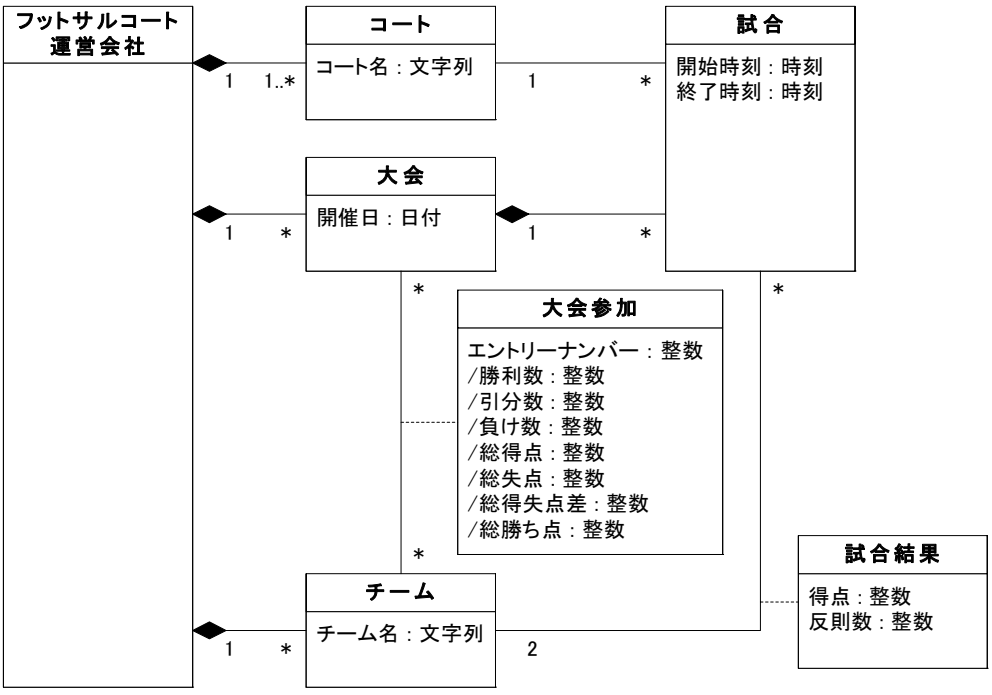

図 7 設問 3 の解答

【発展的内容】
これらの派生属性の導出方法は、クラスの構造を変えることによって改善できる点があります。「図 7 設問 3 の解答」のクラス図では、「チーム」から関連する「試合」群に、その「チーム」が過去に参加した、全ての「大会」の「試合」が含まれています。このため、前述の派生属性を「大会参加」から算出する際に、「大会参加」にひもづく「大会」に関連する「試合」のみを、「試合」群から抽出する必要があります。

この導出方法を改善するには、次の「図 8 試合と大会参加の関連の例」のように、クラス構造を、「試合」と「大会参加」が関連する形に変更します。この変更により、「大会参加」から参照できる「試合」が、対象としている「大会」で行われた「試合」のみに限定されます。

なお、図 8 のクラス図の短所として、関連クラスに対して、更に関連クラスがひもづくため、可読性が低くなる点があげられることに注意してください。

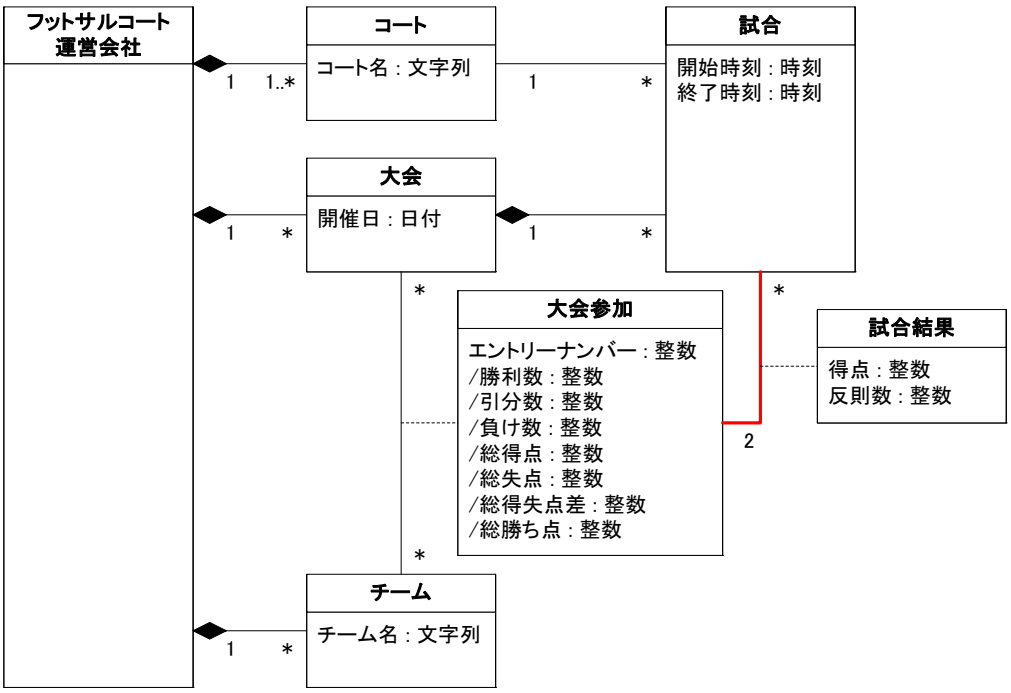

図 8 試合と大会参加の関連の例

分析クラス図での派生属性は、実装レベルのオブジェクトの責務となり得ます。オブジェクトに対する責務割り当ての指針のひとつとして、クレーグ・ラーマンの "GRASP パターン" があります。このパターンのうち、「大会参加」への責務割り当ての際に該当するパターンは、"エキスパート(Expert)" パターンです。このパターンの方針は、"責務の遂行に必要な情報をもっているクラス、すなわち情報エキスパートに責務を割り当てる" です。問題記述の場合、「勝利数」や「総得点」などを算出する責務は、(「大会」と「チーム」毎に存在する)算出のために必要な情報を最も良く知っているべきクラスである、「大会参加」に割り当てます。この観点からも、問題記述のクラス構造よりも、図 8 のクラス構造の方が、「大会」と「チーム」毎の情報が「大会参加」に集中するため、より望ましい構造であると言えます。

なお、このエキスパートパターンを意識することによって、他の GRASP パターン、"疎結合性(Low Coupling)" パターンと "高凝集性(High Cohesion)" パターンの2つのパターンに対しても、良い効果をもたらします。システム全体としてオブジェクト間の依存性が低くなると同時に、複雑さが局所化されることによって、保守しやすく、再利用しやすいクラス構造になります。

また、システムに機能追加が行われていくと、「大会参加」の責務が大きくなりすぎるかもしれません。この場合に参考にできるのは、ロバート・C・マーチンの "オブジェクト指向設計の原則" です。この原則のうち、最もシンプルな原則のひとつである "単一責務の原則(SRP : Single Responsibility Principle)" は、"クラスを変更する理由は1つ以上存在してはならない。" というものです。問題記述の「大会参加」クラスは、チームの大会への参加の仕方を表す概念と、大会で

のチームの成績を表す概念の2つが混在しています。このため、例えば、将来的に、大会に参加するメンバーを管理する変更を「大会参加」に行う際に、成績表の印刷の仕方に変更がない場合でも、成績表に、「大会参加」の変更の影響が及ぶ可能性が高まってしまいます。プログラミング言語によっては、成績表を印刷するモジュールを再コンパイルしなければならないかもしれません。この問題を回避するために、将来的に「大会参加」クラスの責務が肥大した場合や、「大会参加」クラスへの変更が頻繁に発生する場合には、次の「図 9 大会参加と大会成績をクラス分割した例」のように、クラスを分割することを検討します。

なお、図 9 のクラス構造の短所として、クラス数が増えてしまうことや、「大会参加」と「大会成績」の関係が 1 対 1 のため、常に2つのオブジェクトを1つのセットとして扱わなければならなくなる点があげられることに注意してください。

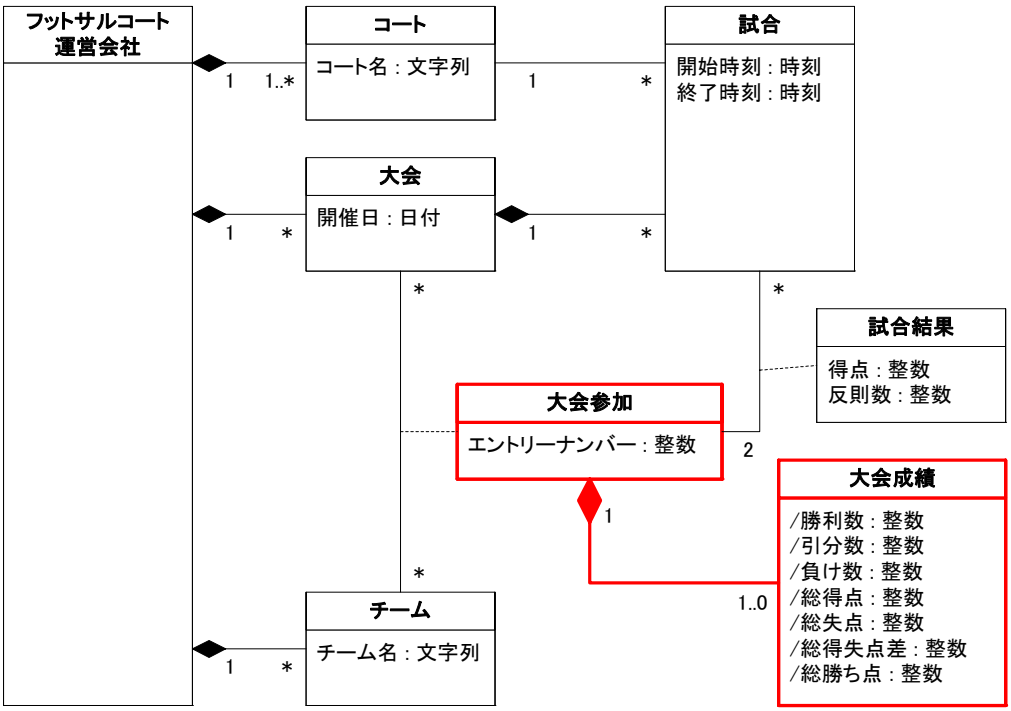

図 9 大会参加と大会成績をクラス分割した例

設問 4

正解

(1)、(3)

解説

(1)を満たすためには、次のような制約を加える必要があります。

- 対象の「試合」が集約される「大会」は、
  その「試合」にひもづく2つの「チーム」からたどれる「大会」の集合に含まれる。

(2)は、クラスの構造によって満たされていると言えます。これは、「試合」から「チーム」への多重度が 2 であるため、「試合」が存在する場合は、必ずチームが 2 つともリンクしていなければならないためです。つまり、「試合」の属性である「開始時刻」「終了時刻」は、チームが確定していないと存在できません。

この条件を、より確実のものにするには、「チーム」側の関連端に {readOnly} 制約を明記します。この制約を明記した場合、1度作成した「試合」の2つの「チーム」は変更されません。

逆に、対戦チームが確定する前に、「試合」の予定をあらかじめ作成しておきたい場合などには、「図 10 試合からチームへの多重度を変更した例」のように、「試合」から「チーム」への多重度を 0 以上 2 以下とすることで対応できます。

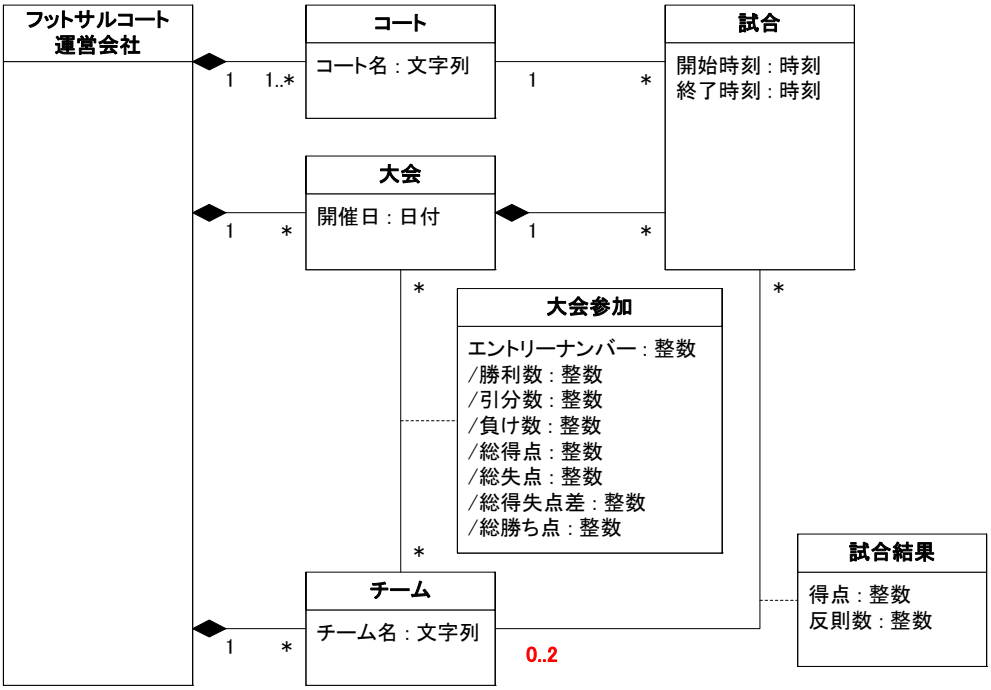

図 10 試合からチームへの多重度を変更した例

(3)は、「試合」と「コート」の間には特に制約が表現されてないため、次のような制約を追加する必要があります。

- 「コート」にひもづく全ての「試合」は、
  同一「大会」に集約される他の「試合」と、「開始時刻」～「終了時刻」が重ならない。

完成形のクラス図の例

最後に、これまでの内容を反映した上で、「成績表」に印字する「大会名」や、フェアプレー賞を印字する際に利用する「総反則数」を加えたクラス図を、例として「図 11 完成形のクラス図の例」に示します。「試合結果」クラスの「勝敗」は、「大会参加」クラスの「勝利数」、「引分数」、「負け数」の算出を助けます。

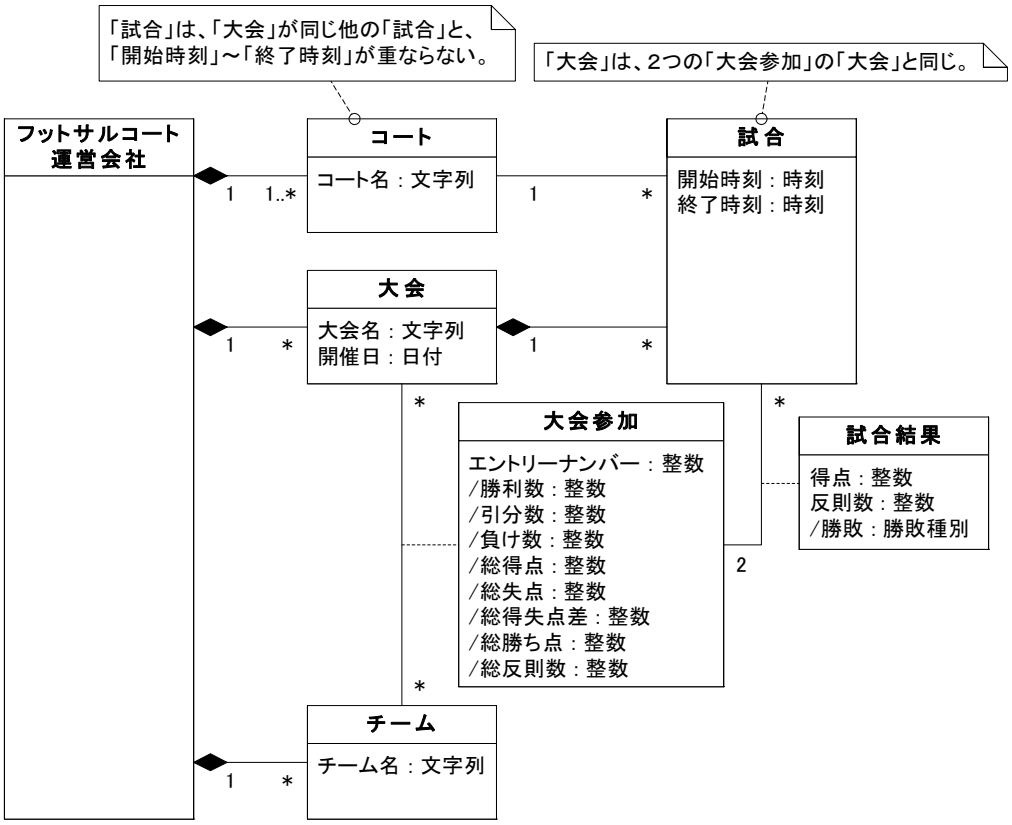

図 11 完成形のクラス図の例

参考文献

- 『実践 UML 第 3 版 オブジェクト指向分析設計と反復型開発入門』
  クレーグ・ラーマン著（ピアソンエデュケーション）
- 『アジャイルソフトウェア開発の奥義 第 2 版 オブジェクト指向開発の神髄と匠の技』
  ロバート・C・マーチン著（ソフトバンククリエイティブ）