1. **Wprowadzenie**.

   Programowanie obiektowe (*ang. object-oriented programming*) jest obecnie najpopularniejszą techniką tworzenia programów komputerowych. Program komputerowy wyraża się jako zbiór **obiektów** będących bytami łączącymi stan (czyli dane) i zachowanie (czyli metody, które są funkcjami operującymi na danych obiektu). W celu realizacji zadania obiekty wywołują nawzajem swoje metody, zlecając w ten sposób innym obiektom odpowiedzialność za pewne czynności. W porównaniu z tradycyjnym programowaniem proceduralnym, w którym dane i procedury nie są ze sobą powiązane, programowanie obiektowe ułatwia tworzenie dużych systemów, współpracę wielu programistów i ponowne wykorzystywanie istniejącego kodu.

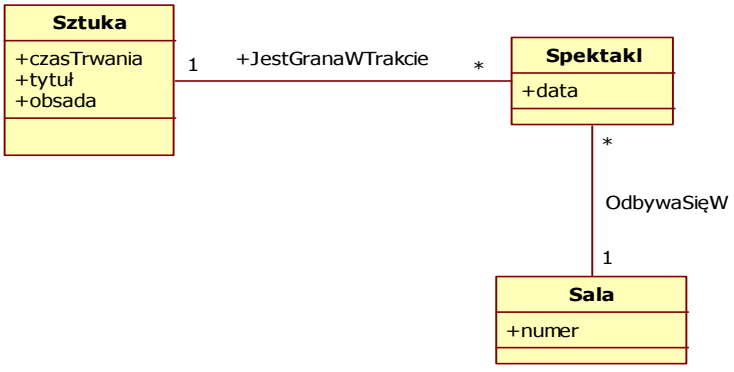
2. **Obiektowe modelowanie dziedziny.**

   Zgodnie ze swoją nazwą **model dziedziny** ma odzwierciedlać pojęcia z modelowanej części świata rzeczywistego oraz ich zależności. W modelu dziedziny nie zajmujemy się **klasami programowymi** (*ang. software class*). Może on posłużyć jako źródło inspiracji przy ich projektowaniu, ale nie w drugą stronę. Tworzenie modelu dziedziny nie jest obowiązkowe ale bardzo przydatne, jeżeli dziedzina problemu, który rozwiązujemy, nie jest dobrze zrozumiała. Jeżeli jest, tworzenie modelu dziedziny należy potraktować jako dekompozycję dziedziny na godne uwagi pojęcia i obiekty. Inspirowanie się nim podczas projektowania systemu pomoże w zmniejszeniu luki reprezentacji.

   W modelu dziedziny pokazuje się:

   - klasy pojęciowe,
   - powiązania między klasami pojęciowymi,
   - atrybuty klas pojęciowych.

   Przykład: częściowy model dziedziny dla aplikacji wspomagającej rezerwację miejsc i sprzedaż biletów w teatrze.



Klasy pojęciowe przedstawiane są w prostokątach. Są to *Sztuka*, *Spektakl* i *Sala*. Każda klasa pojęciowa reprezentuje wszystkie możliwe obiekty jednego typu/rodzaju. Nazwy klas pojęciowych przyjęło się umieszczać w pierwszej przegródce prostokąta i pisać wielką literą. Jeżeli nazwa klasy składa się z kilku członów, łączy się je i każdy kolejny człon również rozpoczyna wielką literą, np. *RozkładJazdy*.

W drugiej przegródce prostokąta reprezentującego klasę pojęciową umieszcza się jej atrybuty. Atrybuty to napisy, liczby, daty bądź wartości logiczne opisujące poszczególne egzemplarze klasy pojęciowej. Każdy egzemplarz/obiekt klasy *Sztuka* posiada na przykład *tytuł* i *czas trwania* oraz *obsadę*.

Uzupełniając klasy o atrybuty trzeba zachować ostrożność. Coś, co na pierwszy rzut oka wydaje się atrybutem, może być inną klasą pojęciową. W naszym przykładzie, mimo że podczas spektakli grane są sztuki, klasa *Spektakl* nie ma takiego atrybutu. Jest za to powiązana z klasą *Sztuka*. Na diagramie jest to pokazane przy pomocy ciągłej linii łączącej obie klasy. Powiązanie oznacza, że między egzemplarzami klas pojęciowych może zachodzić związek, który warto pamiętać przez jakiś czas. Powiązania również mają nazwy. Będziemy je zapisywać tak samo jak nazwy klas pojęciowych, chociaż niektórzy nazwy członów rozdzielają myślnikami. Jako nazw powiązań należy używać zwrotów czasownikowych tak, aby informacje na diagramie tworzyły zdania. W naszym przypadku z diagramu można odczytać:

- **sztuka jest grana w trakcie spektaklu**
- **spektakl odbywa się w sali.**

Domyślnie przyjmuje się kierunek czytania z góry na dół lub od lewej do prawej. Jeżeli niewygodne jest rozmieszczenie klas na diagramie tak, żeby zachować domyślny kierunek, można przy etykiecie umieścić jakiś symbol graficzny wskazujący właściwy kierunek czytania, np. wypełniony na czarno trójkącik.

Podczas obiektowego modelowania dziedziny największą wagę przykłada się do odnajdywania klas pojęciowych i od tego zaczyna się pracę. Następnie rozważa się związki, które mogą wystąpić między egzemplarzami klas pojęciowych i przedstawia je jako powiązania. Odnajdywanie atrybutów jest czymś najmniej istotnym i zostawia się je na koniec.

3. Odnajdowanie klas pojęciowych.

Przy znajdowaniu **klas pojęciowych** (*ang. conceptual class*) należy postępować jak kartograf:

- należy używać istniejących nazw, np. gdy analizujesz system do zbierania wyników w nauce, który ma być używany w liceum, jego użytkownikami będą uczniowie, a gdy jest przeznaczony dla uczelni wyższej to studenci,
- nie zajmować się niczym, co nie dotyczy modelowanej części rzeczywistości; jeżeli pracujesz iteracyjnie powstrzymaj się od rozpoznawania klas pojęciowych, które są nieistotne w obecnej iteracji oraz
- nie dodawać rzeczy, których nie ma.

Analiza fraz rzeczownikowych – innym wygodnym pomysłem na odnajdywanie klas pojęciowych jest analiza fraz rzeczownikowych w tekstowym opisie dziedziny lub wymagań. W przypadku gry w Monopol na pewno warto rozpoznać frazy rzeczownikowe w instrukcji z zasadami gry. Może się wtedy okazać na przykład, że nie uwzględniliśmy kompletów pól, banku, ani kart szans/ryzyka.

Klasy pojęciowe dla gry w monopol:

4. Odnajdywanie powiązań.

Powiązanie (*ang. association*) między klasami wskazuje, że między ich egzemplarzami może występować jakaś zależność. W modelu dziedziny pokazujemy powiązania, które są niezbędne do wypełnienia wymagań informacyjnych i pomagają zrozumieć dziedzinę. Pokazanie zbyt wielu powiązań sprawi, że diagramy będą mało czytelne. Zazwyczaj warto pokazywać powiązania między klasami, jeżeli przez jakiś czas "trzeba pamiętać" o zależności między ich egzemplarzami. Dlatego, mimo że powiązania rysujemy między klasami, myślimy o ich egzemplarzach. Czy trzeba na przykład pamiętać na jakim *Polu* jest *Pionek* albo do jakiego *Gracza* należy? Oczywiście. Jednak informacji, że wartość wskazywana przez *Kostkę* określa, na które *Pole* się ruszyć, nie trzeba przechowywać w modelu. Jest to metainformacja. Tak samo nie ma potrzeby zapamiętywać, że *Gracz* przesuwa *Pionek*.

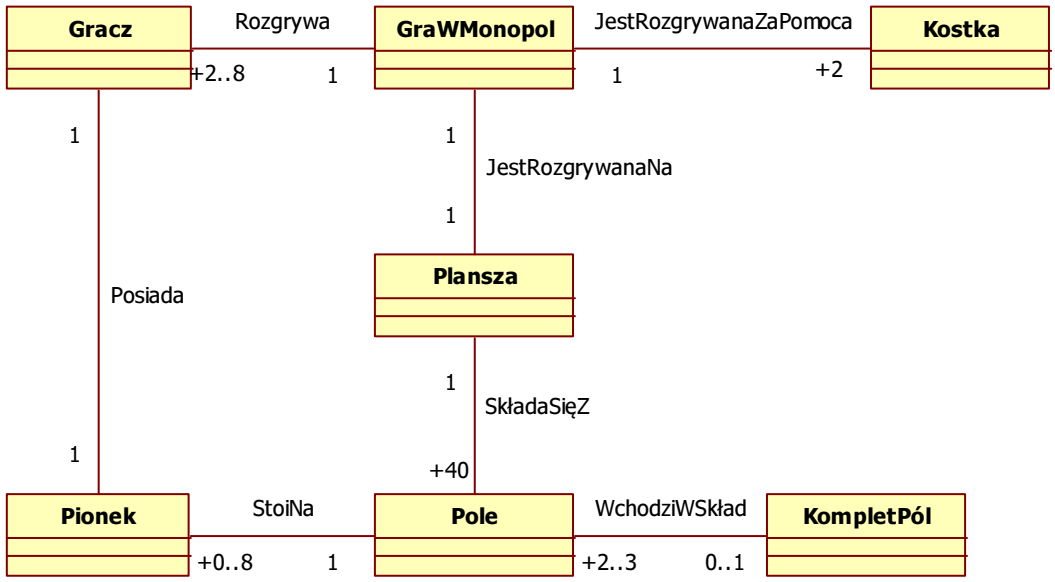
Nazywanie powiązań – nazywanie powiązań nie zawsze jest proste. Wymyślając nazwę sami wiemy co mamy na myśli, ale nie każda nazwa, która pasuje, pozwoli innym zorientować się, o co nam chodziło. Dla wielu powiązań będzie na przykład pasować nazwa *Dotyczy*, ale nie przenosi ona wielu informacji. Pamiętaj o tym nazywając powiązania!

Liczebność – Odnajdując powiązania, powinniśmy zastanowić się nad ich **liczebnością** (*ang. multiplicity*). Liczebność określa, jak wiele egzemplarzy klasy *A* może być powiązane z jednym egzemplarzem klasy *B*. Na diagramach liczebność przedstawia się w postaci wyrażenia umieszczanego obok klasy *A* tuż przy linii obrazującej powiązanie. Przykładowe wyrażenia liczebności to:

- *1* (dokładnie jeden)
- *11* (dokładnie jedenaście)
- *3, 5, 7* (trzy lub pięć lub siedem)
- *2..8* (od dwóch do ośmiu)
- *0..1* (zero lub jeden)
- *1..** (co najmniej jeden)
- * (dowolna ilość również zero)

Określając liczebność pamiętaj, żeby wyrażać ograniczenia związane z dziedziną. Jeżeli na podstawie twojego modelu powstanie w przyszłości system obiektowy, te ograniczenia będą bardzo przydatne i na ich podstawie zostaną określone więzy spójności.

5. Klasy pojęciowe i powiązania dla gry w monopol.

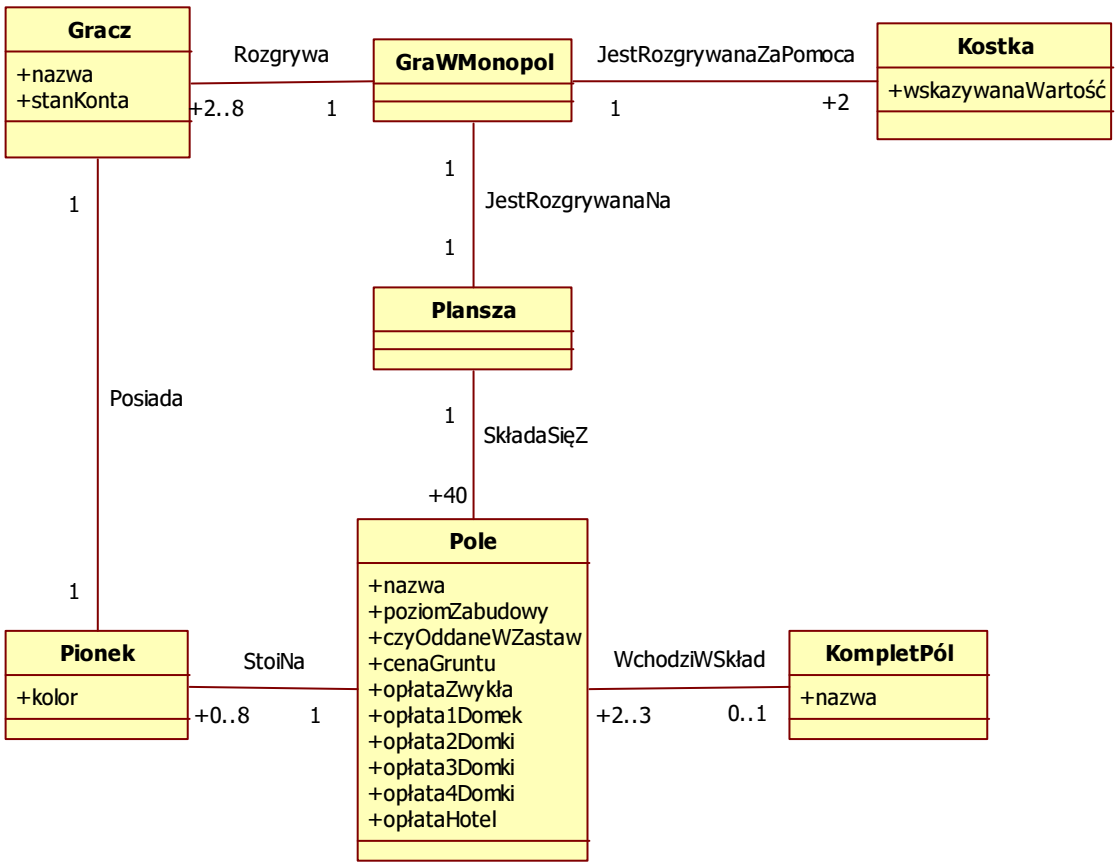


6. Dodawanie atrybutów.

Wartości **atrybutów** (*ang. attirbute*) opisują egzemplarze klas pojęciowych. Nie wszystkie klasy pojęciowe muszą mieć atrybuty. Dodawanie atrybutów jest w zasadzie prostsze niż odnajdywanie klas pojęciowych czy powiązań i zazwyczaj zostawia się je na koniec. Są jednak dwie pułapki.

Po pierwsze, trzeba pilnować, żeby wartości atrybutów rzeczywiście opisywały egzemplarze klas pojęciowych. Może się zdarzyć, że coś, co wydaje się nam atrybutem klasy *A*, powinno tak naprawdę być atrybutem klasy*B*, a między *A* i *B* powinno być powiązanie. Przykładowo *numer* jest atrybutem *Miejsca*, a nie *Biletu*.

Po drugie niech atrybuty będą jedynie wartościami typów podstawowych – "czystymi danymi" – jak napisy, liczby, wartości logiczne czy daty. W przeciwnym przypadku prawdopodobnie znowu lepiej dodać klasę pojęciową i ustanowić z nią powiązanie. Przykładowo kierownik nie jest atrybutem *Teatru*. To oddzielna klasa pojęciowa sama posiadająca wiele atrybutów jak *imię*, *nazwisko* czy *dataUrodzenia*.



Nie jest to jeszcze wersja ostateczna. Nie uwzględniliśmy na przykład kart szansy/ryzyka, a być może w trakcie późniejszych prac (projektowania, kodowania lub dalszych iteracji) zdecydujemy się wprowadzić jakieś zmiany. Nie należy się tym jednak przejmować. Wystarczy, że przez godzinę lub dwie myśleliśmy o dziedzinie zadania. O to właśnie chodzi, żeby unikać "pędu do kodowania" i zastanowić się nad dziedziną. Pamiętaj również żeby pracować iteracyjnie i nie zabierać się za wszystko na raz, tylko zacząć od tego, co najbardziej ryzykowne.

Zadanie:

Wykonaj obiektowe modelowanie dziedziny dla gry w szachy.