

Systemy zdecentralizowane i agentowe - projekt.

Temat: Środowisko do badania różnych rodzajów aukcji

z wykorzystaniem technik agentowych

Autorzy: Tomarz Chmielarz i Andrzej Ryl

[1. Cel projektu](#_gjdgxs)

[2. Zarys architektury](#_30j0zll)

[3. Agenci w systemie](#_1fob9te)

[3.1 AuctionHouse](#_3znysh7)

[3.2 Auction](#_2et92p0)

[3.3 Buyer](#_tyjcwt)

[3.4 DataAnalyzer](#_3dy6vkm)

[3.4.1 Domyślne ustawienia](#_pylr0waoe2ba)

[3.4.2 Ustawienia sterowane za pomocą pliku JSON](#_4x4h9e4086cz)

[4. Typy aukcji](#_1t3h5sf)

[4.1 Aukcja angielska](#_4d34og8)

[4.2 Aukcja japońska](#_2s8eyo1)

[4.3 Aukcja holenderska](#_17dp8vu)

[4.4 Aukcja typu 'Sealed-bid'](#_3rdcrjn)

[4.5 Aukcja eliminacyjna](#_26in1rg)

[5. Strategie kupców](#_lnxbz9)

[5.1 DefaultStrategy](#_35nkun2)

# 1. Cel projektu

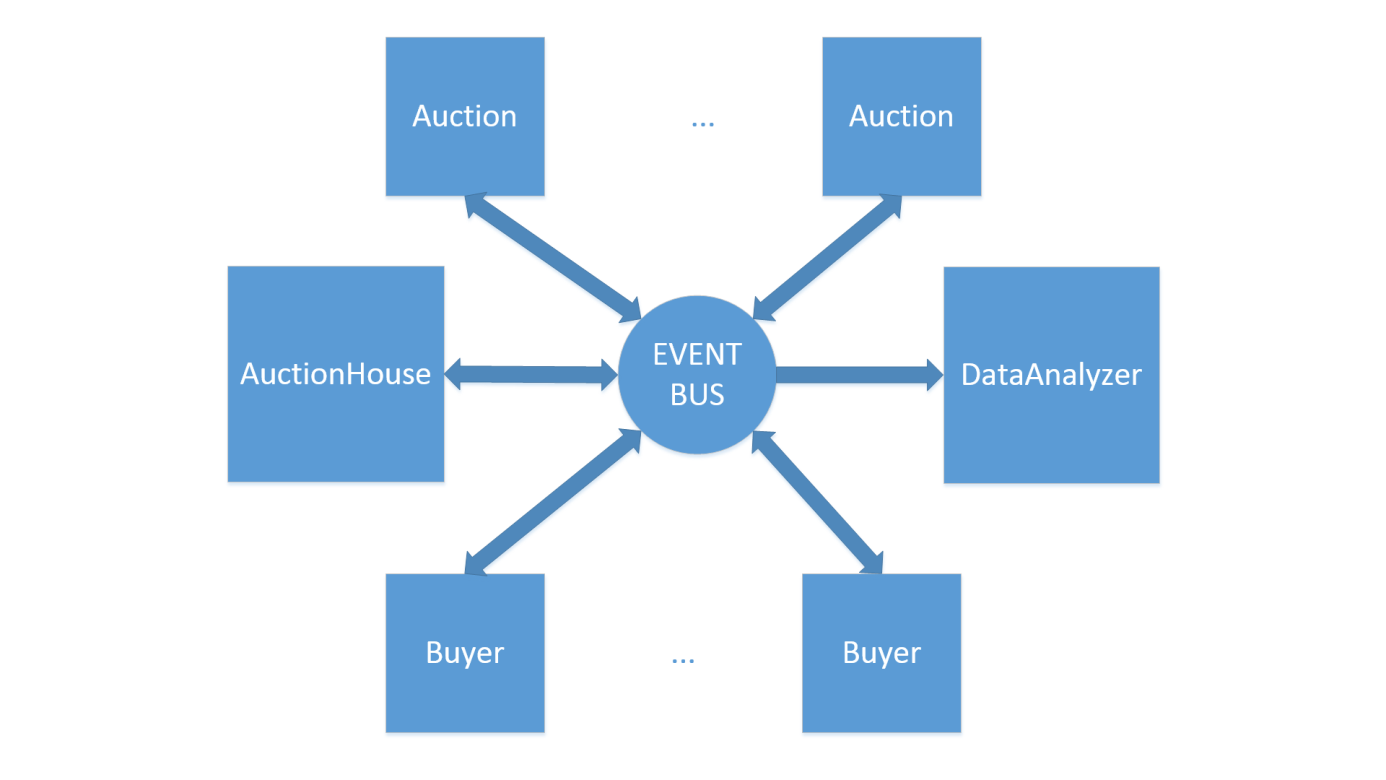
Nasz projekt zakłada stworzenie systemu, który będzie w stanie przetestować różne modele aukcji z pomocą technik agentowych. Do tego celu mamy zamiar użyć agentów reprezentujących kupców o różnych strategiach grania na aukcjach.

W ostatecznej wersji nasz system będzie w stanie porównać skuteczność takich aukcji jak m.in. aukcja angielska, czy aukcja japońska na podstawie zachowań generowanych przez różne strategie kupców. Dzięki temu będziemy w stanie określić, który typ aukcji jest najskuteczniejszy (przynosi największe zyski) dla zdefiniowanych wcześniej warunków (cen początkowych, cen maksymalnych oraz charakterystyki kupujących).

# 2. Zarys architektury

Nasz system opiera się na wzorcu projektowym Publish/Subscribe. Dzięki zastosowaniu tego wzorca, mamy możliwość łatwej komunikacji pomiędzy wszystkimi agentami. Każdy z takich agentów może 'zapisać się' na zdarzenia pewnego typu (np. kupujący zapisują się na zdarzenia typu 'Aukcja nr xxx się rozpoczęła') oraz emitować swoje własne zdarzenia, na które zapisują się inni (np. aukcja emituje zdarzenie mówiące o tym, że się rozpoczęła lub zakończyła).

Ogólną architekturę prezentuje poniższy diagram.



W samym centrum naszego systemu znajduje się komponent EventBus, który służy nam do efektywnej komunikacji między wszystkimi agentami.

Każdy z trzech typów agentów (Auction, Buyer oraz AuctionHouse) emituje swoje własne zdarzenia i 'zapisuje się' na wydarzenia, które go interesują. Dodatkowy agent w systemie (DataAnalyzer) jedynie nasłuchuje wszelkich zdarzeń w systemie i na ich podstawie porównuje skuteczność różnych typów aukcji. Dokładny opis wszystkich wydarzeń zostanie podany w następnych rozdziałach.

Poniżej znajduje się aktualny schemat UML całego systemu:

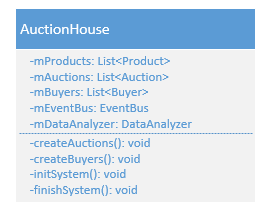


# 3. Agenci w systemie

Nasz system posiada obecnie 4 agentów. Są to:

* AuctionHouse - agent odpowiadający za utworzenie wszystkich komponentów oraz uruchomienie i zakończenie działania całego systemu
* Auction - agent reprezentujący poszczególne aukcje w systemie
* Buyer - agent reprezentujący kupców biorących udział w aukcjach
* DataAnalyzer - agent odpowiedzialny za zebranie wszystkich danych podczas działania systemu, przeanalizowanie ich i na ich podstawie wygenerowanie porównania poszczególnych typów aukcji

## 3.1 AuctionHouse



**Emituje zdarzenia typu:**

* SystemStartedEvent
* SystemFinishedEvent

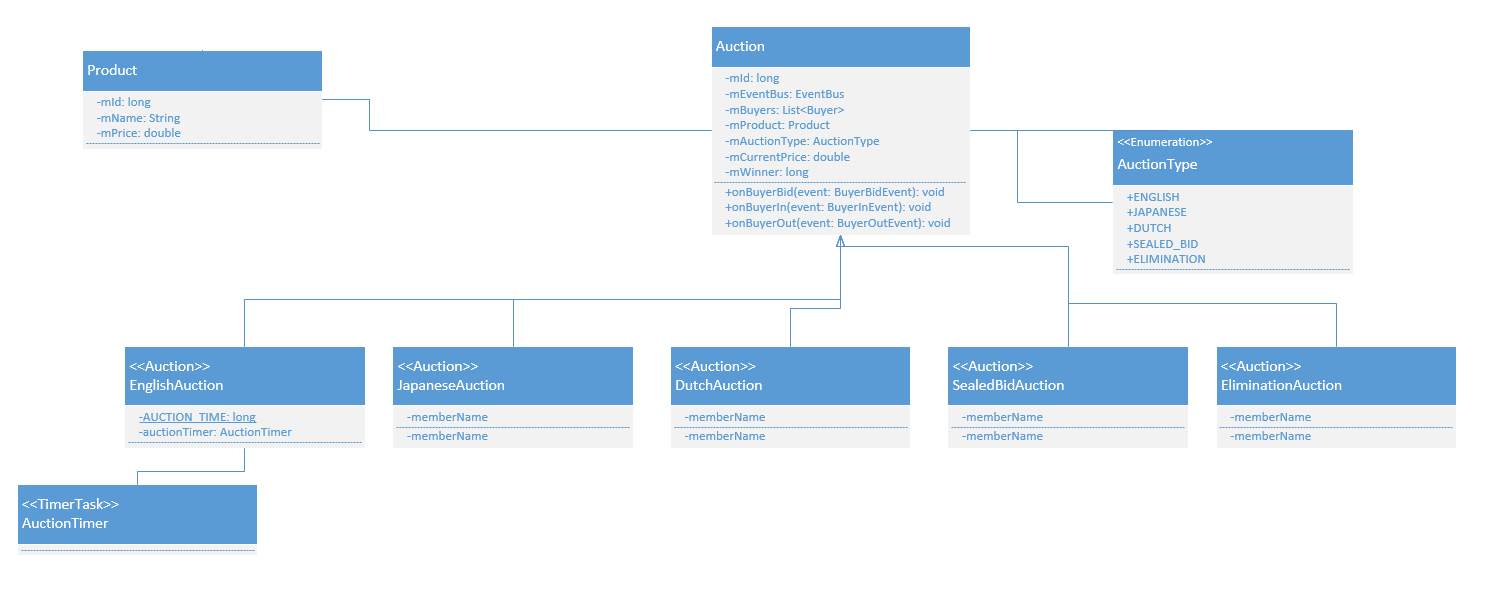
**Nasłuchuje zdarzeń typu:**

* AuctionStartedEvent
* AuctionFinishedEvent

Ten agent odpowiedzialny jest za zainicjowanie oraz zakończenie działania całego systemu. Na początku tworzy on listę aukcji (różnych typów) oraz listę kupujących (korzystających z różnych strategii kupowania) i przyporządkowuje tym aukcjom produkty ze swojej listy (produkty posiadają jedynie nazwę oraz swoją cenę rynkową).

Następnie agent ten inicjuje agenta DataAnalyzer i emituje zdarzenie typu 'SystemStartedEvent', które uruchamia wszystkie aukcje. Po wykonaniu tej czynności agent nasłuchuje zdarzeń typu 'AuctionStartedEvent' (aukcja sygnalizuje, że się rozpoczęła) oraz 'AuctionFinishedEvent' (aukcja sygnalizuje, że się zakończyła). W momencie, w którym wszystkie aukcje się zakończą, emituje on zdarzenie typu 'SystemFinishedEvent' sygnalizujące agentowi DataAnalyzer, że należy przeanalizować wszystkie zebrane dane i wygenerować podsumowanie.

## 3.2 Auction



**Emituje zdarzenia typu:**

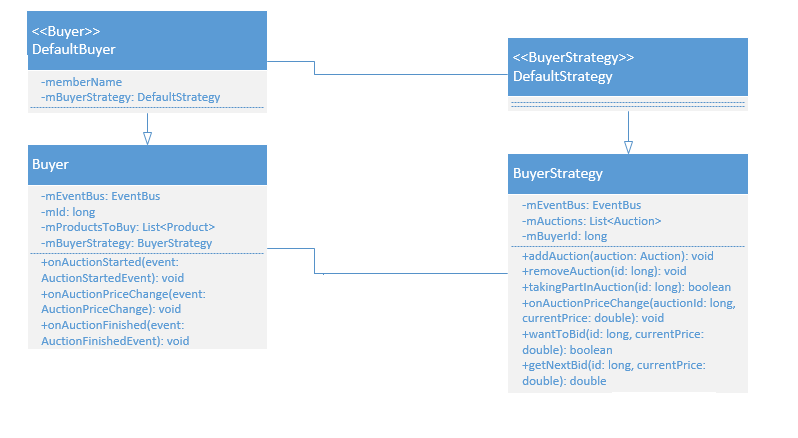
* AuctionStartedEvent
* AuctionFinishedEvent
* AuctionPriceChangeEvent

**Nasłuchuje zdarzeń typu:**

* SystemStartedEvent
* SystemFinishedEvent
* BuyerBidEvent
* BuyerInEvent
* BuyerOutEvent

Ten agent reprezentuje aukcję w systemie. Jest to klasa abstrakcyjna, która umożliwia nam tworzenie różnych typów aukcji korzystających z identycznego interfejsu. Główny model aukcji emituje zdarzenia opisujące jego stan: 'AuctionStartedEvent' (aukcja się rozpoczęła), 'AuctionFinishedEvent' (aukcja się zakończyła) oraz 'AuctionPriceChangeEvent' (zmieniła się cena produktu sprzedawanego na tej aukcji). Wszelkie reakcje na emitowane przez innych agentów zdarzenia są natomiast implementowane w poszczególnych typach aukcji, gdyż zachowania te różnią się w zależności od typu. Poszczególne zachowania zostaną opisane w następnych rozdziałach.

## 3.3 Buyer



**Emituje zdarzenia typu:**

* BuyerBidEvent
* BuyerInEvent
* BuyerOutEvent

**Nasłuchuje zdarzeń typu:**

* AuctionStartedEvent
* AuctionFinishedEvent
* AuctionPriceChangeEvent
* SystemStartedEvent
* SystemFinishedEvent

Ten aktor reprezentuje kupujących biorących udział w aukcjach. Każdy z kupujących korzysta ze swojej własnej strategii i na jej podstawie decyduje, czy przebijać dalej, czy wycofać się aukcji. Podstawowa implementacja kupującego emituje zdarzenia typu 'BuyerInEvent' (kupujący wchodzi do gry), 'BuyerOutEvent' (kupujący wycofuje się z aukcji) oraz 'BuyerBidEvent' (kupujący przebija) na podstawie swojej strategii. Poszczególne implementacje strategii będą opisane w późniejszych rozdziałach.

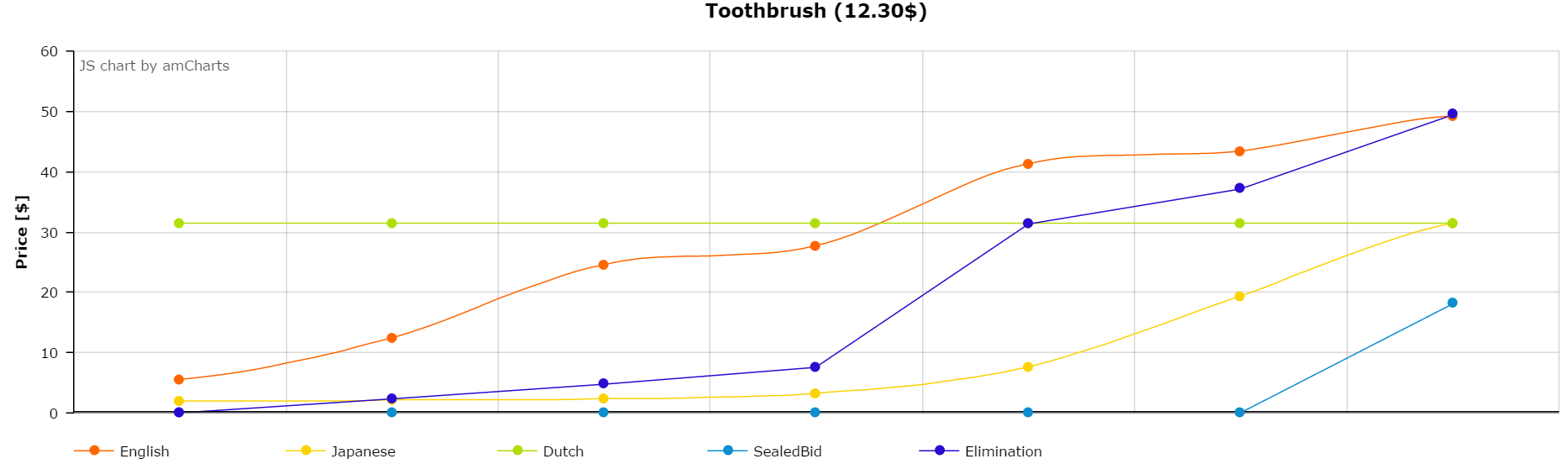
## 3.4 DataAnalyzer

**Ten aktor nasłuchuje wydarzeń związanych ze zmianą ceny i nie wysyła żadnych wydarzeń.**

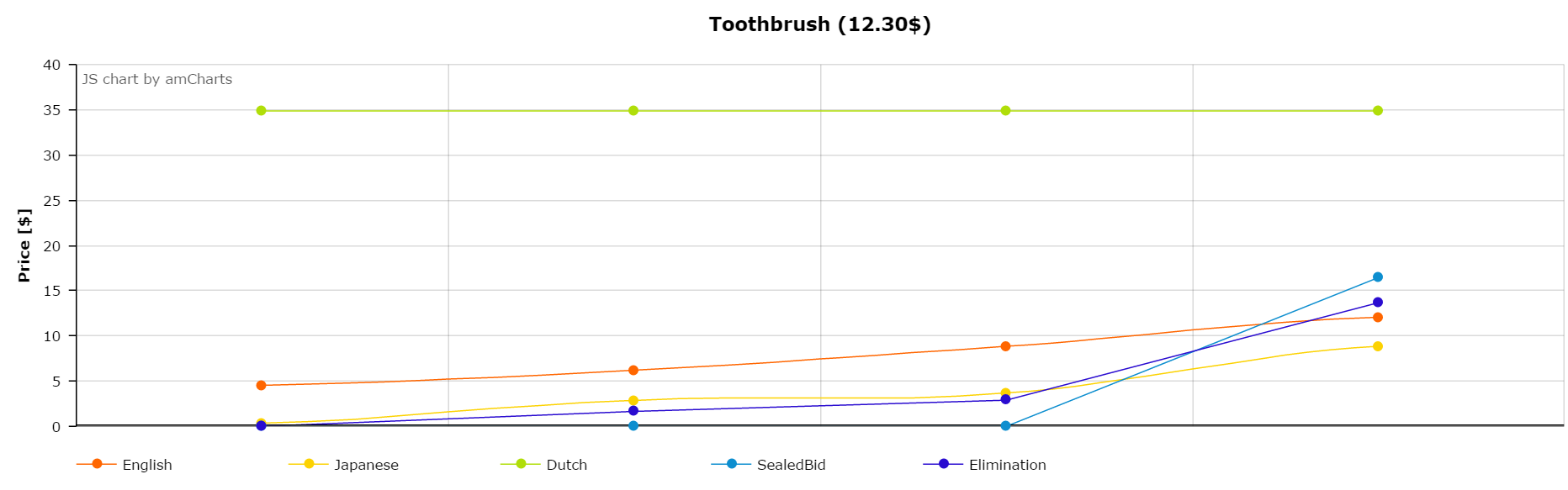
Jego implementacja opiera się o zbieranie informacji odnośnie cen poszczególnych aukcji. Agent AuctionHouse uruchamia po 5 aukcji każdego typu, a później inicjalizuje tego aktora. DataAnalyzer nasłuchuje zdarzeń typu “AuctionPriceChange” i zbiera je w kolekcjach (z podziałem na typ aukcji oraz produkt, który jest na tej aukcji sprzedawany). Następnie - uśredniając wszelkie wyniki - DataAnalyzer buduje wykresy, które prezentują zmieniającą się cenę każdego z produktów. Na każdym grafice (po jednej dla produktu) mamy wykresy związane z każdym z typów aukcji. Dzięki temu możemy z łatwością porównywać skuteczność każdego z typów aukcji.

### 3.4.1 Domyślne ustawienia

Poniżej porównamy wykresy dla czterech produktów



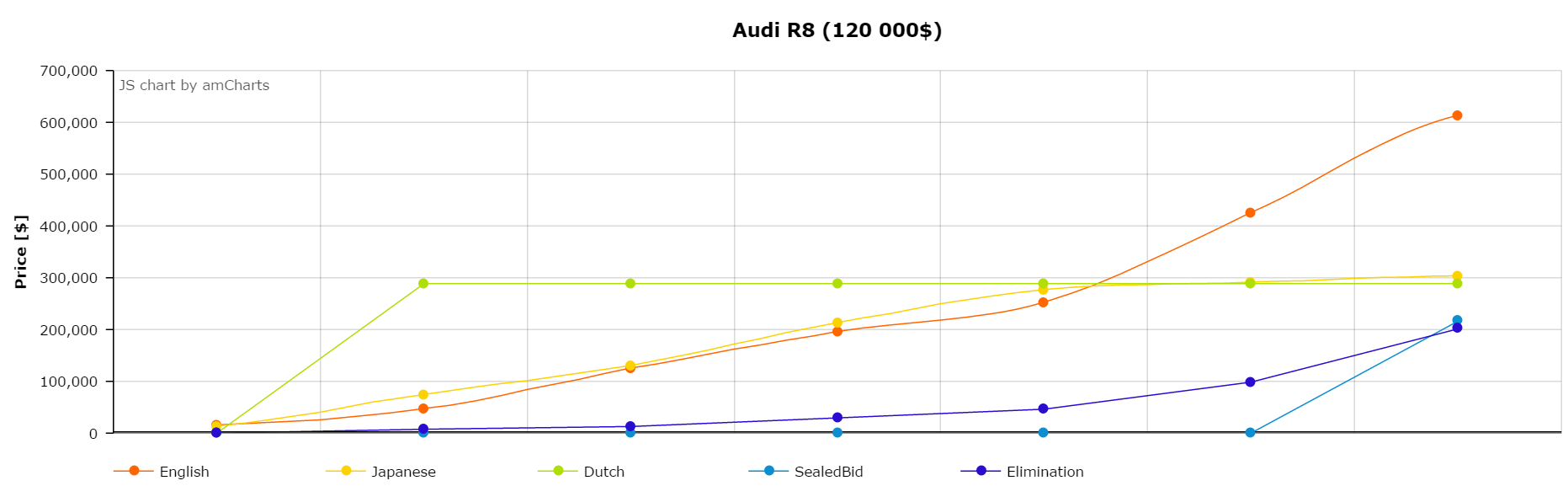
Rys.1. Agresywny kupiec



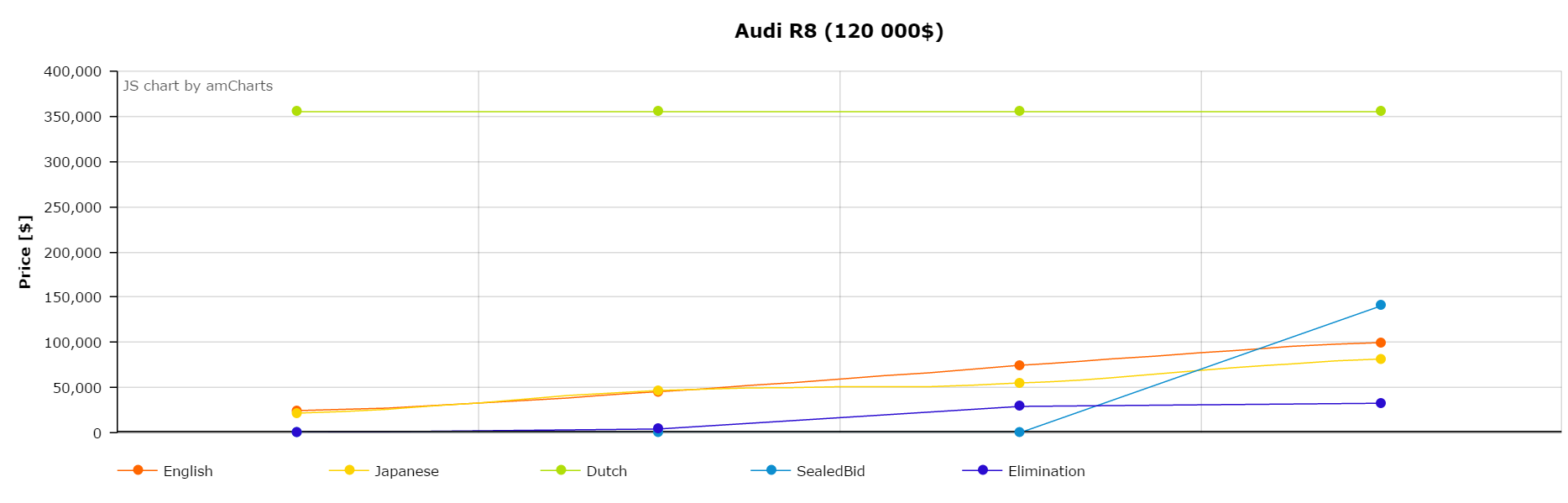
Rys. 2. Domyślny kupiec

Możemy tutaj zauważyć, iż dla agresywnych kupców najskuteczniejsza okazuje się aukcja angielska (jeśli mamy tani produkt tak, jak tutaj). Dla niej ostateczna cena produktu (ok 50$) przekroczyła cenę rynkową (12$) o 300%. Możemy również łatwo zauważyć, iż dla kupców domyślnych nasza implementacja aukcji holenderskiej prowadzi do ciekawego zdarzenia. Kupcy są na tyle elastyczni, że od razu zgadzają się na pierwszą zaproponowaną cenę sprawiając, że ta aukcja staje się najbardziej skuteczna. Na drugim wykresie możemy również zauważyć, iż aukacja typu Sealed-Bid ma tutaj zdecydowanie lepsze przebicie, niż w przypadku gdy w aukcjach brali udział tylko kupujący agresywni.

Poniższy przykład przedstawia to samo porównanie, ale dla droższego produktu.



Rys.3. Agresywni kupcy



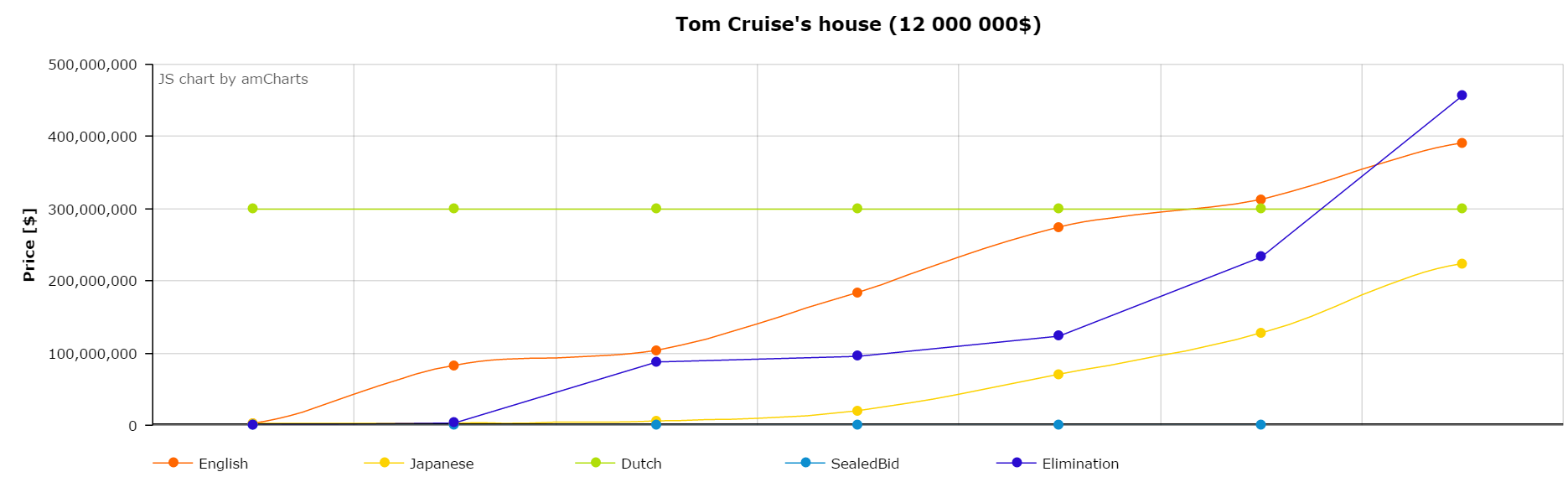
Rys.4. Domyślni kupcy

Tutaj znowu możemy od razu zauważyć elastyczność naszych domyślnych kupców, którzy bez większego zastanowienia zgodzili się na pierwszą cenę zaproponowaną w aukcji holenderskiej.

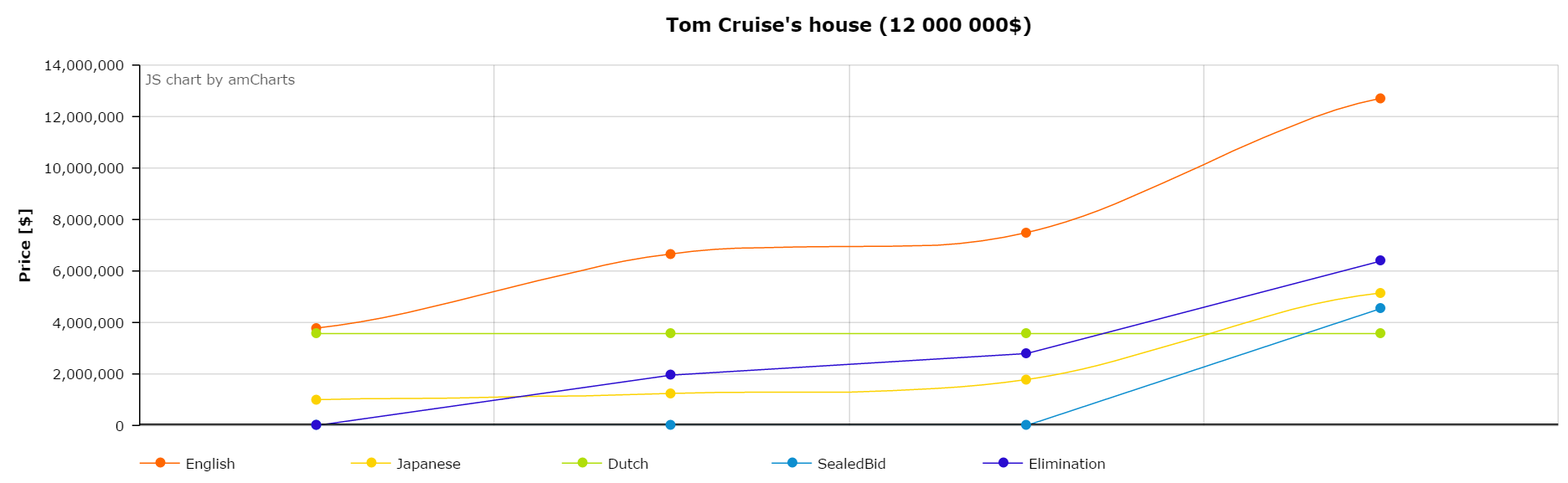
Znów aukcja typu Sealed-Bid okazuje się najlepsza dla kupców domyślnych, a aukcja angielska najlepsza dla kupców agresywnych.

Na tym przykładzie jednak widać zdecydowaną różnicę w cenach końcowych produktów. Dla agresywnych kupców cena ta sięga nawet ponad 600 000$, a w przypadku kupców domyślnych (nie licząc aukcji holenderskiej) tylko 150 000$, co jest różnicą rzędu 400%.

Tę różnicę widać jednak najlepiej w przypadku naszego najdroższego produktu.



Rys.5. Agresywni kupcy



Rys.6 Domyślni kupcy

Jak widzimy w przypadku agresywnych kupców cena końcowa sięga kolosalnej sumy 500 mln $ w porównaniu z tylko 14 mln $ w przypadku kupców domyślnych.

Przy tej cenie jednak możemy zauważyć kilka osobliwości.

Po pierwsze nasza implementacja aukcji holenderskiej już nawet w przypadku kupców domyślnych nie jest najskuteczniejsza. Tam przewyższają ją wszystkie inne typy aukcji z aukcją angielską na czele.

Warto jednak również zwrócić uwagę na fakt, iż w przypadku tak wysokiej ceny i kupujących agresywnych aukcja angielska przestała być najskuteczniejsza. Wyprzedziła ją (o dobre 100mln $) aukcja eliminacyjna.

### 3.4.2 Ustawienia sterowane za pomocą pliku JSON

W tym punkcie na podstawie najdroższego produktu porównamy wykresy zmiany cen w zależności od wartości każdego z trzech parametrów sterujących ceną, które możemy ustawić w pliku konfiguracyjnym. Są to:

**bid\_factor -** ten współczynnik steruje wielkością ofert proponowanych przez kupców. Jego wartość może przyjmować liczbę od 0 do 1, gdzie 0 oznacza brak ofert, a 1 oznacza najbardziej hojne oferty

**japanese\_auction\_change\_factor -** ten współczynnik steruje wielkością zmiany cen w przypadku japońskiej aukcji (cena zmienia się tam odgórnie, a nie za sprawą kupujących)

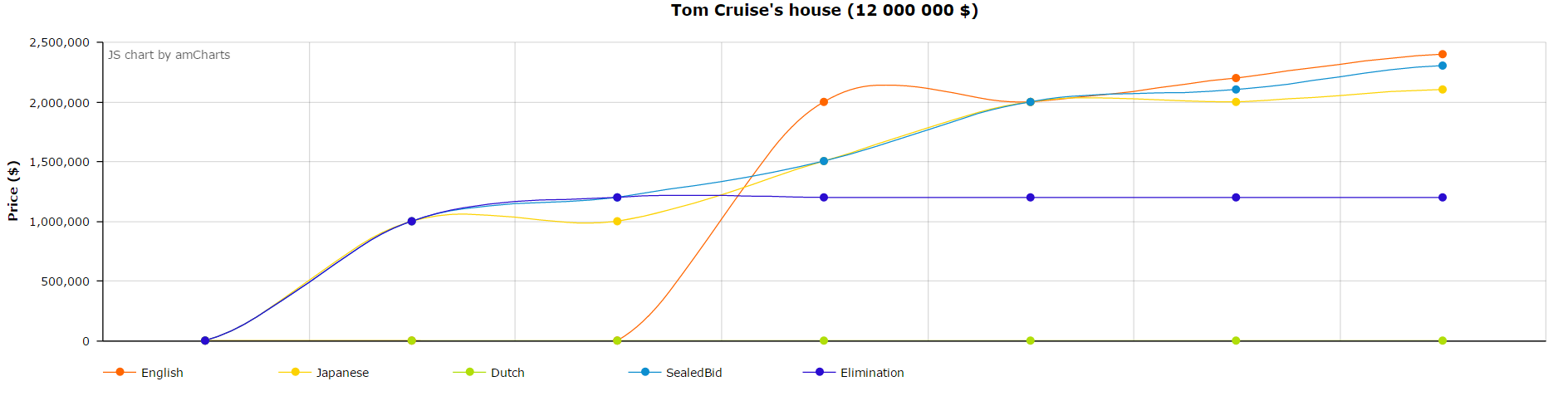
**wants\_to\_bid\_factor -** ten współczynnik określa jak daleko są w stanie się posunąć kupujący. Im współczynnik wyższy, tym większe prawdopodobieństwo, że kupujący podbiją ofertę

Na poniższych wykresach przedstawione są sytuacje skrajne, żeby zademonstrować jaki wpływ na działanie systemu mają powyższe parametry. Na wykresach w poprzednim punkcie wartości tych parametrów były uśredniane, żeby uzyskać jak najbardziej rzeczywisty obraz.

**bid\_factor** 0.1

**japanese\_auction\_change\_factor** 0.1

**wants\_to\_bid\_factor** 0.1

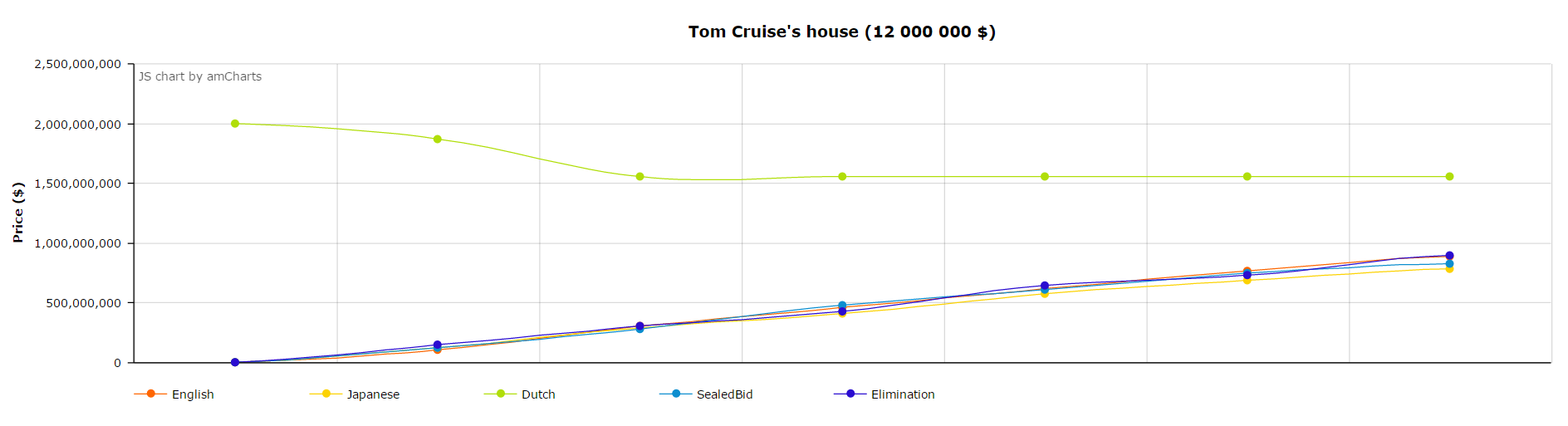


Jak widzimy z powyższego wykresu takie ustawienie parametrów znacząco wpłynęło na wysokość uzyskanej kwoty. Wykresy z poprzedniego podpunktu sięgały 500 000 000 $ przy agresywnych kupcach oraz 14 000 000 $ przy domyślnych kupcach. Przy takich ustawieniach cena sięgnęła zaledia 2 500 000 $. Można też bardzo łatwo zauważyć małe zmiany w cenie pomiędzy kolejnymi 'rundami' aukcji. Na to wpłynął bardzo niski współczynnik wants\_to\_bid oraz bid\_factor. Kupcy bardzo rzadko decydowali się na podbicie ceny, a jeżeli już to robili to proponowana kwota była bardzo mała.

**bid\_factor** 0.9

**japanese\_auction\_change\_factor** 0.9

**wants\_to\_bid\_factor** 0.9

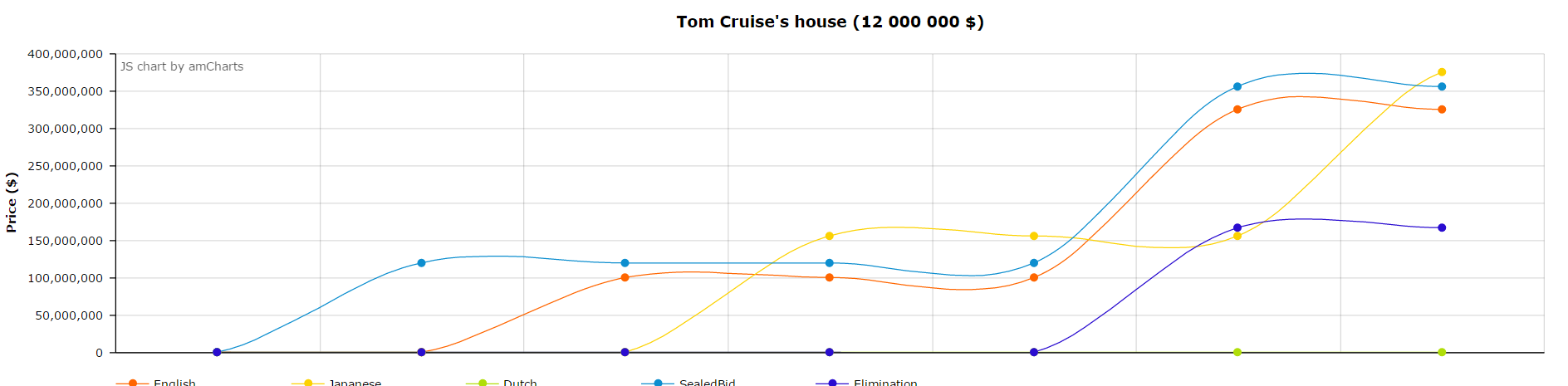


Pierwszą rzeczą, która najbardziej rzuca się w oczy jest fakt, iż w końcu holenderska aukcja okazała się być najbardziej skuteczna. Jej forma (na początku prowadzący aukcję podaje astronomiczną cenę, a potem stosunkowo ją obniża, aż któryś kupiec postanowi ją przyjąć) wykorzystała fakt, iż - dzięki wysokim parametrom wants\_to\_bid oraz bid\_factor - kupcy byli skłonni bardzo szybko zgodzić się na bardzo wysoką ceną. Pozostałe aukcje szły praktycznie tym samym tempem, gdyż kupcy właściwie za każdym razem decydowali się na podbicie ceny i robili to prawie zawsze maksymalną możliwą wartością.

**bid\_factor** 0.9

**japanese\_auction\_change\_factor** 0.9

**wants\_to\_bid\_factor** 0.1

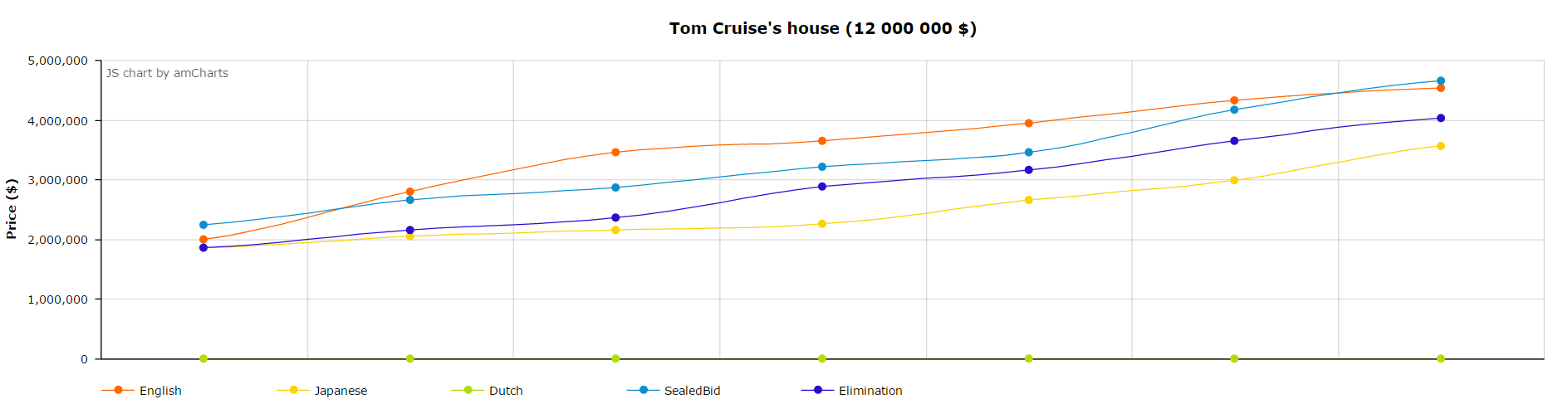


Pierwsza rzecz, którą można na tym wykresie zauważyć jest fakt, że holenderska aukcja znów się nie udała. Przez bardzo niski współczynnik wants\_to\_bid\_factor żaden z kupców nie zdecydował się na zalicytowanie w tej aukcji. Wpływ ten współczynnika widać również doskonale w aukcji eliminacyjnej. Dopiero pod sam koniec pojawiła się pierwsza oferta, która utrzymała się do końca aukcji. Tak niski współczynnik wants\_to\_bid sprawia, że zmiana ceny następuje bardzo rzadko i chociaż kupcy przebijają o bardzo wysokie kwoty (współczynnik bid\_factor na poziomie 0.9) to aukcje i tak dotarły tylko do 400 000 000 $, podczas gdy aukcje gdzie wszystkie współczynniki były ustawione na 0.9 dotarły to cen rzędu 1 500 000 000 $.

**bid\_factor** 0.1

**japanese\_auction\_change\_factor** 0.1

**wants\_to\_bid\_factor** 0.9



Na powyższym wykresie znów możemy zauważyć, że aukcja holenderska nie udała się. Kupcy bardzo często chcieli brać udział w aukcji (dzięki parametrowi wants\_to\_bid ustawionemu na 0.9) ale parametr bid\_factor ustawiony na 0.1 nie pozwalał im na to. Pozostałe aukcje znów idą tym samym torem - w każdej 'rundzie' pojawia się nowa oferta, ale jest ona bardzo mała. Przez to żadna z aukcji nie przekroczyła nawet 5 000 000 $.

Z powyższych wykresów możemy wysnuć jeden bardzo ważny wniosek. Przy ustawieniu parametrów na warunki skrajnie brzegowe prawie we wszystkich przypadkach typ aukcji nie ma tak dużego znaczenia. Jedynie w przypadku ustawienia wszystkich parametrów na 0.9 mogliśmy zauważyć zdecydowaną przewagę aukcji holenderskiej. W pozostałych przypadkach aukcje szły równym torem i osiągały podobne efekty.

# 4. Typy aukcji

Każdy z typów aukcji rozszerza podstawowego agenta, jakim jest Auction. Podstawowa implementacja obsługuje cztery zdarzenia :

* AuctionStartedEvent - to zdarzenie jest emitowane podczas rozpoczynania aukcji
* BuyerBidEvent - jeśli cena przekazana w tym zdarzeniu jest wyższa od aktualnej, ta zmienia się, a agent emituje zdarzenie typu 'AuctionPriceChangedEvent'
* BuyerInEvent - jeśli ten kupujący jeszcze nie bierze udziału w naszej aukcji, to agent go dodaje
* BuyerOutEvent - jeśli kupujący bierze udział w naszej aukcji, to agent go usuwa

Poszczególne typy jedna wprowadzają szereg zmian w celu implementacji reguł, które rządzą poszczególnymi aukcjami.

## 4.1 Aukcja angielska

**Zasady**

* sprzedawca daje cenę początkową
* kupcy starają się przebić cenę
* aukcja kończy się po sprecyzowanym wcześniej czasie
* zwycięzca MUSI kupić produkt po tej cenie, jaką podał

**Zmiany w podstawowej implementacji**

* podczas rozpoczynania aukcji uruchamiany jest timer (wykorzystywana jest do tego klasa AuctionTimer)
* podczas rozpoczynania aukcji generowana jest cena produktu (losowa wartość mniejsza od ceny rynkowej)
* każda wyższa propozycja ceny powoduje wyznaczenie nowego tymczasowego zwycięscy
* po zakończeniu czasu emitowane jest zdarzenie typu 'AuctionFinishedEvent'. W tym samym czasie tymczasowy zwycięzca zostaje ostatecznym zwycięzcą

## 4.2 Aukcja japońska

**Zasady**

* sprzedawca daje cenę początkową
* kupujący mówią czy są "in", czy nie, czyli czy są skłonni kupić produkt po tej cenie, czy nie
* następnie sprzedawca zwiększa cenę i znowu pyta
* jeśli kupiec odpada, to nie może wejść z powrotem
* aukcja kończy się, gdy zostaje jeden kupiec i on MUSI kupić

**Zmiany w podstawowej implementacji:**

* podczas rozpoczynania aukcji ustawiany jest timer odliczający czas do do następnego momentu związanego z podniesieniem ceny (służy do tego dodatkowy typ eventu IncreasePriceEvent)
* podczas rozpoczynania aukcji generowana jest losowa cena początkowa na podstawie ceny rynkowej produktu
* po rozpoczęciu aukcja rozsyła informację o swoim starcie a następnie o tym że cena została zmieniona a kupujący mogą się zapisywać
* po każdorazowym zakończeniu się czasu oznaczającego wzrost ceny aukcja weryfikuje ilu użytkowników było w tym czasie “in” na zakup w cenie ustalonej w poprzedniej iteracji:
  + Jeśli uczestników będących “in” było więcej niż jeden, aukcja zwiększa swoją cenę (o wylosowaną wartość) i inicjuje timer na nowo oraz informuje o zmianie ceny, kupujący znowu mogą przesłać informacje o tym czy dalej są “in”
  + Jeśli był tylko jeden uczestnik “in” oznacza to że aukcja się zakończyła i został wyłoniony zwycięzca, rozsyłana jest informacja o zakończeniu
  + Jeśli nie było żadnego uczestnika to aukcja zakończyła się bez zwycięzcy i rozsyła informację o zakończeniu

## 4.3 Aukcja holenderska

**Zasady**

* sprzedawca daje bardzo wysoką cenę
* potem stosunkowo ją zmniejsza
* jak kupiec usłyszy zadowalającą go cenę, to się zgłasza
* ten, który pierwszy się zgłosi, MUSI kupić produkt

**Zmiany w podstawowej implementacji:**

* podczas rozpoczynania aukcji ustawiany jest timer odliczający czas do do następnego momentu związanego z obniżeniem ceny (służy do tego dodatkowy typ eventu DecreasePriceEvent)
* podczas rozpoczynania aukcji generowana jest losowa bardzo wysoka cena początkowa na podstawie ceny rynkowej produktu
* po rozpoczęciu aukcja rozsyła informacje o zmianie ceny i oczekuje pierwszego kupującego
* po każdorazowym zakończeniu się czasu oznaczającego spadek ceny aukcja wysyła do Siebie informację o zmniejszeniu ceny, jest ona zmniejszana o losową wartość a następnie wysyłana jest informacja o zmianie ceny na którą zareagować mogą kupujący
* aukcja kończy się w momencie w którym pierwszy z kupujących w odpowiedzi na zmianę (zmniejszenie) ceny prześle informacje o chęci zakupu (BuyerInEvent) - zostaje wtedy ustalony zwycięzca (first-come-first-served) a aukcja zostaje zakończona

## 4.4 Aukcja typu 'Sealed-bid'

**Zasady**

* każdy z kupców daje tzw. 'sealed-bid' czyli swoją cenę, ALE potajemnie
* żaden z kupców nie zna propozycji pozostałych kupców
* kupiec z najwyższą ofertą MUSI kupić produkt ale po cenie jaką sam podał lub jaką podał kupiec, który był na drugim miejscu

**Zmiany w podstawowej implementacji:**

* podczas rozpoczynania aukcji uruchamiany jest timer odliczający czas w którym kupujący mogą składać swoje “tajne oferty”
* po rozpoczęciu aukcja rozsyła informacje o tym że kupujący mogą się zapisywać (zapis dostępny jest tylko w czasie od rozpoczęcia do zakończenia odliczania)
* aukcja otrzymując informację o podbiciu ceny przez kupującego (de facto złożeniu ofert) zapisuje te informacje do listy aby na końcu mieć możliwość wyłonienia zwycięzcy
* po upływie czasu trwania aukcji wyłaniany jest zwycięzca:
  + jeśli ofert było więcej niż jedna są one sortowane według wartość a następnie właściciel najwyższej oferty zostaje zwycięzcą z wartością drugiej najwyższej oferty (normalnie ma on opcję wyboru jednakże oczywistym jest fakt że wybierze niższą ofertę)
  + jeśli była tylko jedna oferta jej właściciel zostaje zwycięzcą z ceną którą zaoferował
  + jeśli nie było żadnych ofert (np. nikt nie był zainteresowany produktem) aukcja zostaje zakończona bez zwycięzcy

## 4.5 Aukcja eliminacyjna

**Zasady**

* rozpoczynamy aukcję typu sealed-bid
* kupiec z najniższą ofertą odpada, a jego cena staję się jawną ceną minimalną drugiej rundy aukcji typu sealed-bid
* kontynuujemy tak, aż zostanie jeden kupiec i on wtedy płaci cenę minimalną ostatniej rundy

**Zmiany w podstawowej implementacji:**

* podczas rozpoczynania aukcji uruchamiany jest timer odliczający czas w którym mogą zapisywać się do uczestnictwa w aukcji eliminacyjnej
* po rozpoczęciu aukcji rozsyłana jest informacja że kupujący mogą się zapisywać, w tym czasie przesyłają oni informacje że są “in”, później zacznie się etap eliminacyjny
* po zakończeniu czasu zapisywania aukcja oczekuje ofert od kupujących którzy są “in” - czeka aż każdy z nich złoży ofertę
* kiedy wszyscy kupujący złożą ofertę (sprawdzane po każdej przesłanej ofercie) aukcja wyłania uczestnika z najniższą ofertą - zostaje on wykluczony z listy uczestników “in” odpowiednim komunikatem (KickedOutEvent) a następnie aukcja przesyła komunikat o zmianie ceny i cały proces powtarza się od nowa aż do momentu gdy po eliminacji “najsłabszego” uczestnika zostanie tylko jeden kupujący który jest “in”

# 5. Strategie kupców

Każdy z kupców w naszym systemie działa według swojej strategii. Istnieje ogólny model reprezentowany przez klasę 'BuyerStrategy', który jest rozszerzany przez poszczególne implementacje. Podstawowy model na podstawie implementacji metod 'getNextBid' oraz 'wantsToBid' emituje zdarzenia typu 'BuyerBidEvent' (przebicie ceny), 'BuyerInEvent' (kupujący bierze udział w aukcji) oraz 'BuyerOutEvent' (kupujący wycofuje się z aukcji).

## 5.1 DefaultStrategy

Ta strategia jest strategią najprostszą i została stworzona głównie po to, aby testować pozostałe komponenty. Kupujący przebija cenę zupełnie losowo (losowo wybierana jest zarówno kwota jak i fakt samej chęci przebicia).

**Implementacja wantsToBid**

Kupujący losowo wybiera, czy chce przebijać, czy nie.

**Implementacja getNextBid**

Kupujący losowo wybiera kwotę z przedziału (1.2 \* obecna cena; 1.3\*obecna cena).

# 6. Sterowanie parametrami - JSON

W związku z tym że projekt zakłada stworzenie środowiska służącego do badania efektywności różnych rodzajów aukcji musi ono być konfigurowalne. W tym celu stworzony został moduł konfiguracyjny odpowiadający z wczytywanie danych dotyczących poszczególnych elementów symulacji tj. produktów, aukcji, kupujących (i stałych odpowiadających za zachowania).

Komponent pozwala na wczytywanie poszczególnych obiektów na podstawie dostarczonego do programu pliku typu JSON. Możemy w nim definiować następujące elementy:

* **Produktów:**
  + **id** - id produktu potrzebne do połączenia produktu z aukcjami
  + **name** - nazwa produktu
  + **price** - średna wartość rynkowa
* **Aukcji:**
  + **auctionType** - określa typ aukcji english/japaness/dutch/seal bid/elimination
  + **productId -** określa produkt które dotyczy aukcja
* **Kupujących:**
  + **productIds** - lista produktów których kupujący będzie poszukiwał na aukcjach
  + **minDecisionRate** - określa minimalną szansę na podjęcie “pozytywnej decyzji”
  + **bidRate** - określa poziom “zmiany” ceny przez kupującego (czyli jak bardzo kupujący przy kolejnych ofertach będzie zwiększał swoją ofertę)

