**AUTOMAT-BILETOWY:**

Wszystko opisane tym kolorem jest bardzo skrótowym wyjaśnieniem, co powinien robić dany segment kodu.  
 **1. Zdarzenie występujące przy zmianie ilości biletów**

Jeżeli wartośc mieści się w przedziale to dodaj/odejmij bilet, w przypadku kiedy cena do zapłaty po odjęciu biletu jest mniejsza niż 0 zwróć wrzucone monety.  
spinner\_N\_20.addChangeListener(new ChangeListener() { public void stateChanged...

Zdarzenie to wyłapuje każdą zmianę jaka wystąpi, kiedy użytkownik albo program zmieni ilość biletów i wykonuje następujące czyności:  
  
- Do zmiennej „value” przypisana jest obecna wartość spinnera.  
-Sprawdź czy wartość jest >9, jeżeli tak to ustaw na 9 (nie chcemy 10) -Sprawdź czy wartość jest <0, jeżeli tak to ustaw na 0 (nie chcemy -1)

Jeżeli wartość mieści się w przedziale to sprawdź, czy wartość się zmniejszyła lub zwiększyła od ostatnego stanu. Zmienna „stateOfSpinners” jest tablicą przechowującą wszystkie poprzednie stany spinnerów odpowiednio indeksy tablicy 0-5 do spinnerów 0-5.

Jeżeli wartośc się zwiększyła to wywałaj metodę „addTicket” (Patrz 12)

Jeżeli wartość się zmniejszyła to: wywołaj metodę „deleteTicket” (Patrz 13), jeżeli metoda zwróciła prawdę wywołaj metodę „giveBackMoney” (Patrz 9), „resetSpinners” (Patrz 6), „setText” (Patrz 7), zaktualizuj wartość value.

Przypisz do elementu tablicy stan do zapamiętania, który przy następnej zmianie wartości tego Spinnera stanie się „poprzednim stanem”. Wywołaj „setText” (Patrz 7).

**2. Zdarzenie występujące przy kliknięciu przycisku dodania biletu**Dodaj bilet do automatu, jeżeli po „wrzuceniu pieniędzy” cena jest mniejsza niż 0 wydaj bilety z resztą i wszystko wyzeruj.

A001zlButton.addActionListener(new ActionListener(){public void action Performed…

Zdarzenie wyłapuje to każde kliknięcie:

-Wywołaj metodę „isHereAnyMoney” (Patrz 5), jeżeli zwróci true, to wywołaj metodę „addMoney” (Patrz 8), jeżeli zwróci true to wywołaj „getTickets” (Patrz 4) , wywołaj „resetSpinners” (Patrz 6) i na koniec wywołaj funkcję „setText” (Patrz 7).

**3. Zdarzenie występujące przy kliknięciu przycisku anuluj**

Zwróc wrzucone monety i wszystko wyzeruj

anulujButton.addActionListener(new ActionListener(){public void action Performed…

wywołaj: „giveBackMoney” (Patrz 9), resetSpinners (Patrz 6), setText (Patrz 7).

**4. Funkcja getTicket**

Funkcja wypisuje ile biletów wydrukowano na podstawie stanów spinnerów .

**5. Funkcja isHereAnyMoney**

Jeżeli cena do zapłaty jest równa 0 to wypisz „Nie można dodac biletu” zwroc false, w przeciwnym razie zwróć true.

**6. Funkcja resetSpinners**

Ustaw wartość wszystkich spinnerów na 0. Warto zauważyć, że na samym końcu ustawiamy cene do zapłąty na 0, jest to ważne, ponieważ przy odejmowaniu biletów (zmiana z 5 na 0) algorytm odejmie 5 biletów i może wystąpić ujemny koszt.

**7. Funkcja setText**

Zaktualizuj ilość pieniędzy do zapłaty na ekranie (jeżeli nie ma nic wyświet nic).

**8. Metoda addMoney**

Odejmij „wrzuconą monetę” od ceny do zapłaty. WAŻNE: metoda musi coś zwracać, żeby „powiedzieć” przyciskom, czy można już drukować bilety, czy nie.

Utwórz nowy obiekt Money, odejmij wartość monety od ceny do zapłaty i zaokrąglij tę cenę, i jeżeli jest <= 0 wywołaj metodę „return\_money” (Patrz 11), ustaw cenę na 0 i zwróc true, w przeciwnym razie zwróc false.

**9. Metoda giveBackMoney**

Zwróc wszystkie monety, które zostały zapisane w tablicy, i wyczyść tablicę monet.

**10. Metoda setTicketsToZero**

Ustaw cenę do zapłaty na 0.

**11. Metoda returnMoney**

Wyczyść tablicę monet (nie jest już nam potrzebna, bo wydajemy resztę, a nie pieniądze już zwrócone), wydawaj pięniądze, starając się wydawać jak największe nominały cały czas zaokrąglając cene do wydania.

**12. Metoda addTicket**

Dodaj cenę do zapłaty dla wywołanego indeksu (odpowiednio dla każdego spinnera), na koniec zaokrąglij cenę.

**13. Metoda deleteTicket**

Odejmij cenę do zapłaty dla wywołanego indeksu (odpowiednio dla każdego spinnera). Zaokrąglij cenę i na koniec zwróc true kiedy cena po odjęciu jest <0 w przeciwnym razie zwróc false.

**14. Metoda round**

Zaokrąglij cenę do zapłaty do 2 miejsc po przecinku.

**15. Metoda getCostOfTickets**

Zwróc cenę do zapłaty.

**16.Podsumowanie:**

Widzimy, że automat wyda pieniądze w następujących przypadkach:

- Dodamy dokładnie tyle samo pięniędzy/więcej ile jest do zapłaty (zwróci resztę i bilety).

-Kiedy pieniądze są w automacie i będziemy odejmować pięniądze aż do momentu, w którym cena do zapłaty jest niższa niż ilość pięniędzy jakie wcześniej wrzuciliśmy (odda te same pieniądze co wrzuciliśmy) (cała ta operacja zostanie potraktowana jako błąd).

-Zostanie wciśnięty przycisk anuluj (odda te same pieniądze co wrzuciliśmy).

Klasa value reprezentuje wartość (klasa abstrakcyjna, bo wartość sama w sobie jest abstrakcją).

Klasa Money reprezentuje już obiekt „Pieniądz” i dziedziczy po klasie Value (kompozycja)

Klasa Machine tworzy i niszczy „Pieniądze” (klasę Money) (Asocjacja)

Klasa MyGui odpowiada za wyłapywanie zdarzeń i wysyłanie ich do obiektu „Machine”, który tworzy w konstruktorze (Agregacja)