Laboratorium 2 (praca własna 14.10 i 16.10 + konsultacje MS Teams)

Przed przystąpieniem do rozwiązania koniecznie przeczytać:
https://bulldogjob.pl/articles/1194-how-not-to-break-your-app-with-hashcode-and-equals

1. *(Prezentacja rozwiązania zadania 7* z zajęć 1)*
2. Utwórz nowy projekt w IntelliJ i dodaj do niego klasę *Main* z metodą *main*.
3. Dodaj do projektu klasę X oraz klase Y dziedziczącą po klasie X (`class Y extends X { … }`).
4. W klasie X dodaj pola `protected typu int xMask` (od razu zainicjalizuj wartością `0x00ff`) oraz `fullMask`;
5. Zdefiniuj w klasie X konstruktor domyślny, w którym przypiszesz do *fullMask* wartość xMask.
6. Zdefinij w klasie *Y* składową *yMask* i zainicjalizuj ją od razu przy definicji wartością 0xff00;
7. Zdefiniuj w klasie *Y* konstruktor, w którym przypiszesz do *fullMask*:
   `fullMask |= yMask;`
8. W klasie *Main* dodaj w metodzie main deklarację zmiennej y typu Y a następnie utwórz nowy obiekt *Y* poprzez `new Y();`
9. Prześledź przy pomocy debugger'a i pracy krokowej wartości przyjmowane przez *xMaska, yMask, fullMask* w kolejnych krokach wywołania konstruktora.
10. Dodaj w klasie X oraz Y poniższą metodę (w klasie Y następnie zmień znak „&" na „|") a następnie wywołaj ja w metodzie kontruktora (raz X a raz Y):

```
xMask = mask(xMask); //(dla X)
i
yMask = mask(xMask); //(dla Y).

public int mask(int orig) {
      return (orig & fullMask);
}
```

11. Spróbuj odpowiedzieć sobie na pytania:
    a. Jaka implementacja metody jest wywoływana przy wywołaniu metody.
    b. Czy jeśli metoda korzysta z pól tego obiektu czy mamy gwarancję, że pola są odpowiednie zainicjalizowane.
12. Stwórz klasę Dog z dwoma polami: name and says. W metodzie main( ), utwórz 2 obiekty d1, d2 o wartości pola name "rex" (mówiący "Ruff!") i "scruffy" (mówiący "Wurf!"). Następnie wyświetl ich imiona I to co mówią.
13. Utwórz referencje do nowego obiektu Dog o nazwie d3 i przypisz do niej obiekt d1. Następnie utwórz nowy obiekt d4 o wartości pola name = „rex" i says = „Ruff!". Porównaj dla wszystkich kombinacji działanie porównania poprzez == oraz equals( ) (wszystkie 4 referencje: d1, d2, d3, d4).
14. Napisz metody equals i hashCode dla klasy Dog (poeksperymentuj z różnymi implementacjami) i jeszcze raz przetestuj porównywanie poprzez equals i ==. Przed implementacją przeczytaj poniższy artykuł.
15. Odpowiedz sobie na pytanie:
    a. Jaka jest różnica w domyślnym działaniu equals/hashCode a Twoimi implementacjami?
16. Napisz klasę reprezentująca talię kart: Deck. Klasa Card reprezentująca kartę powinna mieć pola typu odpowiedniego wyliczeniowego: rank oraz suit. Napisz metodę „fabryki" zwracające posortowaną talię kart oraz metodą shuffle zwracającą potasowaną talię kart na bazie dostarczonej talii kart.
17. Dopisz program, który ma 4 graczy i każdemu graczowi „rozdaje" 5 kart z potasowanej talii kart a następnie je wyświetla.
18. Do klasy Card dopisz metody equals i hashCode (wygeneruj generatorem IntelliJ a następnie pobaw się różnymi implementacjami).

19. Potestuj działanie metod equals/hashCode poprzez wsadzanie obiektów Card do kontenera typu HashSet.

UWAGA: punkty od 15 do 18 przydadzą Ci się do realizacji zadania nr 2