Pytania z kartkówek apohllo

Kartkówka 1.

1. Abstrakcja to
   (a) **możliwość definiowania nowych typów**
   (b) ukrywanie implementacji
   (c) nadpisywanie metod z klasy nadrzędnej
   (d) traktowanie funkcji jako typu danych
2. Która spośród wymienionych cech charakteryzuje obiektowe podejście do programowania
   (a) **abstrakcja**
   (b) leniwa ewaluacja
   (c) metaprogramowanie
   (d) statyczna typizacja
3. Która spośród wymienionych cech charakteryzuje obiektowe podejście do programowania
   (a) **dziedziczenie**
   (b) statyczna typizacja
   (c) metaprogramowanie
   (d) kompilacja
4. Wybierz stwierdzenie, które jest prawdziwe w odniesieniu do konstruktora
   (a) **moze byc prywatny**
   (b) musi posiadac argumenty wywolania
   (c) musi zwracac wartosc za pomoca slowa return
   (d) jego nazwa moze byc inna niz nazwa klasy
5. Która spośród wymienionych cech nie charakteryzuje obiektowego podejścia do programowania
   (a) **leniwa ewaluacja**
   (b) hermetyzacja
   (c) dziedziczenie
   (d) abstrakcja
6. Dziedziczenie wiąże się bezpośrednio z
   (a) **możliwością nadpisywania metod**
   (b) traktowaniem funkcji jak typów danych
   (c) ukrywaniem implementacji
   (d) metaprogramowaniem

7. Hermetyzacja to

   (a) **ukrywanie implementacji**
   (b) możliwość definiowania nowych typów danych
   (c) nadpisywanie metod z klasy nadrzędnej
   (d) traktowanie funkcji jak typu danych

8. Wybierz stwierdzenie, które nie jest prawdziwe w odniesieniu do konstruktora:

   (a) **musi być publiczny**
   (b) jego nazwa musi być taka sama jak nazwa klasy
   (c) nie musi posiadać argumentów
   (d) nie może zwracać wartości

9. Wybierz stwierdzenie, które nie jest prawdziwe w odniesieniu do konstruktora

   (a) **jego nazwa nie musi być identyczna jak nazwa klasy**
   (b) może być prywatny
   (c) może posiadać argumenty wywołania
   (d) nie może zwracać wartości

Kartkówka 2.

1. Które spośród wymienionych porównań da w wyniku wartość `true`? Przyjmij, że `Color` to typ wyliczeniowy posiadający wartość `Red`, a `Person` to klasa nie implementująca żadnej metody

   (a) **`Color.Red == Color.Red;`**
   (b) `new Integer(200) == new Integer(200);`
   (c) `String a = "a"; String b = "b"; a + b == "ab";`
   (d) `Person a = new Person(); Person b = new Person(); a.equals(b);`

2. Które spośród wymienionych porównań da w wyniku wartość `true`? Przyjmij, że `Person` to klasa nie implementująca żadnej metody.

   (a) **`"ab" == "ab";`**
   (b) `new Integer(200) == new Integer(200);`
   (c) `1e-25 * 1e25 == 1;`
   (d) `new Person().equals(new Person());`

3. Które porównanie da w wyniku wartość `true`

   (a) **`Arrays.equals(new String[]{"a"}, new String[]{"a"});`**
   (b) `new String[]{"a"}.equals(new String[]{"a"});`
   (c) `new String[]{"a"} == new String[]{"a"}`
   (d) `Arrays.Equals(new String[]{"a"}, new String[]{"a"});`

4. Które porównanie da w wyniku wartosc `true`

   (a) **`Arrays.deepEquals(new String[]{"a"}, new String[]{"a"});`**

   (b) `Arrays.Equals(new String[]{"a"}, new String[]{"a"});`

   (c) `Array.equal(new String[]{"a"}, new String[]{"a"});`

   (d) `new String[]{"a"}.equals(new String[]{"a"});`

5. Które spośród wymienionych porównań da w wyniku wartość `true`? Przyjmij, że `Person` to klasa nie implementująca żadnej metody.

   (a) **`Math.abs((1e-25 * 1e25) - 1)` $<$ `1e-6;`**

   (b) `new String[]{"a"} == new String[]{"a"};`

   (c) `new Person().equals(new Person());`

   (d) `new Integer(200) == new Integer(200);`

6. Które spośród wywołań jest niepoprawne (przyjmij, że `Color` to typ wyliczeniowy posiadający wartości `Red` i `Blue`)

   (a) **`1.equals(1);`**

   (b) `new Integer(1).equals(1);`

   (c) `"a".equals("b");`

   (d) `Color.Red.equals(Color.Blue);`

7. Które spośród wymienionych porównań da w wyniku wartość `true`? Przyjmij, że `Person` to klasa nie implementująca żadnej metody

   (a) **`new Integer(10) == new Integer(10);`**

   (b) `new Person().equals(new Person());`

   (c) `a = "a"; b = "b"; a + b == "ab";`

   (d) `1e-25 * 1e25 == 1;`

8. Która sygnatura jest poprawną sygnaturą metody służącej do porównywania obiektów

   (a) **`public boolean equals(Object other)`**

   (b) `public boolean equal(Class other)`

   (c) `public Boolean Equals(Object other)`

   (d) `public boolean equals(Class other)`

9. Wartości w tablicy o deklaracji `Integer[20]`

   (a) **są zainicjowane jako `null`**

   (b) ich odczyt powoduje wystąpienie wyjątku `NullPointerException`

   (c) są zainicjowane wartościami losowymi

   (d) są zainicjowane jako `0`

10. Wybierz poprawne zdanie

    (a) **w Javie tylko wartość `true` ewaluowana jest jako prawda**

    (b) w Javie wartość `1` ewaluowana jest jako prawda

    (c) w Javie pusty łańcuch ewaluowany jest jako fałsz

    (d) w Javie pusta tablica jest ewaluowana jako fałsz

11. W Javie przekroczenie zakresu wartości dostępnych dla danego typu (np. `long`)
    (a) **nie jest sygnalizowane w żaden sposób**
    (b) jest sygnalizowane przez kompilator
    (c) skutkuje wystąpienie wyjątku `RuntimeError`
    (d) skutkuje wystąpieniem wyjątku `RangeError`
12. Wskaż prawdziwe zdanie dotyczące języka Ruby:
    (a) **wartość `false` jest ewaluowana jako prawda**
    (b) pusty łańcuch ewaluowany jest jako fałsz
    (c) pusta tablica ewaluowana jest jako fałsz
    (d) wartość `nil` ewaluowana jest jako prawda

Kartkówka 3.

1. Pole oznaczone w klasie `A` modyfikatorem `protected`
    (a) **jest dostępne dla metod należących do klas dziedziczących z klasy `A`**
    (b) jest dostępne tylko dla metod należących do klasy `A`
    (c) jest dostępne dla metod należących do klas, należących do pakietu, w którym zdefiniowana jest klasa `A`
    (d) jest dostępne tylko dla obiektu do którego należy
2. Wskaż nieprawdziwe zdanie dotyczące pola, które w klasie A nie ma żadnego modyfikatora
    (a) **jest niepoprawną konstrukcją w języku Java**
    (b) posiada odrębną wartość dla każdego obiektu należącego do klasy `A`
    (c) może być odczytywane w metodach klasy `A`
    (d) może być zapisywane w metodach klas należących do tego samego pakietu co klasa `A`
3. Pole oznaczone w klasie `A` modyfikatorem `final`
    (a) **może być zapisywane w konstruktorze klasy `A`**
    (b) może być zapisywane w metodach należących do klas dziedziczących z klasy `A`
    (c) może być zapisywane w metodach należących do klasy `A`
    (d) może być odczytywane w dowolnej metodzie
4. Pole, które w klasie `A` nie posiada żadnego modyfikatora
    (a) **może być zapisywane w metodach klas należących do tego samego pakietu co klasa `A`**
    (b) jest niepoprawną konstrukcją w języku Java
    (c) może być zapisywane w dowolnej metodzie
    (d) może być odczytywane w dowolnej metodzie

5. Pole oznaczone w klasie `A` modyfikatorem `private`
   (a) **jest dostępne dla wszystkich metod należących do klasy `A`**
   (b) jest dostępne w metodach klas dziedziczących z klasy `A`
   (c) jest dostępne dla metod należących do klas należących do pakietu, w którym zdefiniowana jest klasa `A`
   (d) jest dostępne tylko dla obiektu do którego należy
6. Wskaż nieprawdziwe zdanie dotyczące pola oznaczonego w klasie `A` jako `static`
   (a) **posiada odrębną wartość dla każdego obiektu należącego do klasy `A`**
   (b) może być zapisywane we wszystkich metodach klasy `A`
   (c) posiada jedną wartość dla wszystkich obiektów należących do klasy `A`
   (d) może być odczytywane we wszystkich metodach klasy `A`
7. Pole oznaczone w klasie `A` modyfikatorem `static`
   (a) **posiada jedną wartość dla wszystkich obiektów należącego do klasy `A`**
   (b) może być zapisywane tylko w metodach statycznych klasy `A`
   (c) może być odczytywane tylko w metodach statycznych klasy `A`
   (d) posiada odrębną wartość dla każdego obiektu należącego do klasy `A`
8. Wskaż nieprawdziwe zdanie dotyczące pola oznaczonego w klasie `A` jako `private`
   (a) **może być odczytywane w metodach należących do klas dziedziczących z klasy `A`**
   (b) posiada odrębną wartość dla każdego obiektu należącego do klasy `A`
   (c) może być zapisywane w metodach należących do klasy `A`
   (d) może być odczytywane w metodach należących do klasy `A`
9. Wskaż nieprawdziwe zdanie dotyczące pola oznaczonego w klasie `A` jako `protected`:
   (a) **może być odczytywane w metodach należących do klas nadrzędnych względem `A`**
   (b) może być zapisywane w metodach należących do klas dziedziczących z klasy`A`
   (c) może być odczytywane w metodach należących do klas dziedziczących z klasy `A`
   (d) może być zapisywane w metodach należących do klasy `A`

Kartkówka 4.

1. Metoda oznaczona modyfikatorem `protected`
   (a) **jest wirtualna**
   (b) nie może być ponownie zdefiniowana w klasach dziedziczących
   (c) może być wywoływana tylko przez inne obiekty danej klasy
   (d) nie ma dostępu do pól prywatnych w klasie, w której jest zdefiniowana

2. Metoda oznaczona modyfikatorem `static`
   (a) **nie ma dostępu do pól instancyjnych**
   (b) może wywoływać metody instancyjne
   (c) nie może być wywoływana w metodach instancyjnych
   (d) nie ma dostępu do pól statycznych
3. Metoda abstrakcyjna to:
   (a) **metoda nie posiadająca implementacji**
   (b) każda metoda, która jest zdefiniowana w klasie abstrakcyjnej
   (c) metoda, która może występować tylko w interfejsie
   (d) metoda, która posiada domyślną implementacje
4. Metoda oznaczona modyfikatorem `final`
   (a) **nie może być ponownie zdefiniowana w klasie dziedziczącej**
   (b) służy do odczytu pól oznaczonych modyfikatorem `final`
   (c) zwraca wartości, które nie mogą być modyfikowane
   (d) nie może być zdefiniowana w klasie nadrzędnej
5. Metoda wirtualna to metoda:
   (a) **której zachowanie zależy od rzeczywistego typu obiektu**
   (b) której zachowanie zależy od deklarowanego typu obiektu
   (c) która nie może modyfikować stanu instancji danej klasy
   (d) która nie posiada implementacji
6. Słowo kluczowe `super`
   (a) **służy do wywołania metody w klasie nadrzędnej**
   (b) może pojawić się tylko w definicji metody chronionej
   (c) służy do wskazania klasy nadrzędnej w definicji klasy
   (d) nie może pojawić się w konstruktorze
7. Metody zdefiniowane w interfejsie
   (a) **są abstrakcyjne jeśli nie posiadają modyfikatora `default`**
   (b) są publiczne tylko jeśli posiadają modyfikator `public`
   (c) muszą być zdefiniowane w wszystkich klasach implementujących ten interfejs
   (d) nie są wirtualne
8. Klasa abstrakcyjna to
   (a) **klasa, w której występują metody abstrakcyjne**
   (b) klasa, w której występują pola abstrakcyjne
   (c) klasa, w której wszystkie występujące metody są abstrakcyjne
   (d) każda klasa w języku Java

9. Metoda oznaczona modyfikatorem `default`
    (a) **definiuje domyślna implementację metody w interfejsie**
    (b) występuje wyłącznie w klasach abstrakcyjnych
    (c) nie zawiera implementacji
    (d) może być wywoływana tylko przez klasy znajdujące się w tym samym pakiecie
10. Wskaż nieprawdziwe stwierdzenie dotyczące metody oznaczonej modyfikatorem `private`
    (a) **nie może być ponownie zdefiniowana w klasie dziedziczącej**
    (b) nie może być wywoływana przez obiekty innych klas
    (c) nie jest wirtualna
    (d) może być wywoływana przez inne obiekty tej samej klasy

Kartkówka 5.

1. Blok `try...catch` służy do
    (a) **obsługi wyjątków**
    (b) deklarowania wyjątków zgłaszanych przez metodę
    (c) zgłaszania wyjątków
    (d) sterowania przepływem wykonania metod
2. Użycie instrukcji `try` "z zasobami"
    (a) **wymaga od zasobu implementacji interfejsu `AutoCloseable`**
    (b) wymaga użycia sekcji `catch`
    (c) nie jest zalecane dla najnowszych programów
    (d) polega na zwolnieniu zasobu w sekcji `finally`
3. Słowo kluczowe `throw` służy do
    (a) **zgłaszania wyjątku**
    (b) sterowania przepływem wykonania metod
    (c) przechwytywania wyjątków
    (d) deklarowania wyjątków, które są zgłaszane przez metodę
4. Blok `finally`
    (a) **wykona się niezależnie od wystąpienia wyjątku**
    (b) wykona się tylko jeśli wystąpi wyjątek
    (c) nie wykona się jeśli nie wystąpi wyjątek
    (d) może występować przed blokiem `catch`
5. Wskaz nieprawdzie stwierdzenie dotyczące wyjątków typu `unchecked exception`
    (a) **nie mogą być definiowane przez użytkownika**
    (b) dziedziczą z klasy `Exception`
    (c) brak ich obsługi nie musi być zgłaszany przez metodę
    (d) mogą nie być przechwytywane w żadnej metodzie

6. Zwolnienie zasobu
   (a) **powinno nastąpić w bloku `finally`**
   (b) może wystąpić w dowolnym bloku: `try`, `catch`, `finally`
   (c) powinno nastąpić w bloku `try`
   (d) powinno nastąpić w bloku `catch`
7. Wskaż nieprawdziwe stwierdzenie dotyczące wyjątków typu `checked exception`
   (a) **dziedziczą z klasy `RuntimeException`**
   (b) brak ich przechwycenia musi być deklarowane przez metody
   (c) mogą być definiowane przez użytkownika
   (d) mogą nie być przechwycone w żadnej metodzie
8. Wskaż nieprawdzie stwierdzenie dotyczące wyjątków typu `unchecked exception`
   (a) **nie mogą być definiowane przez użytkownika**
   (b) mogą nie być przechwytywane w żadnej metodzie
   (c) brak ich obsługi nie musi być zgłaszany przez metodę
   (d) dziedziczą z klasy `Exception`
9. Słowo kluczowe `throws` służy do
   (a) **deklarowania wyjątków zgłaszanych przez metodę**
   (b) zgłaszania wyjątku
   (c) obsługi wyjątków
   (d) sterowania przepływem w metodach

Kartkówka 6.

1. Wyjątek `UnsupportedOperationException` występuje
   (a) **jeśli wywołamy metodę, która nie jest obsługiwana przez daną implementację**
   (b) jeśli w iteratorze skończą się elementy do iterowania
   (c) jeśli przekażemy niepoprawną wartość do metody (np. `null`)
   (d) jeśli wywołamy metodę na obiekcie `null`
2. Która klasa nie implementuje interfejsu `Collection`?
   (a) **`LinkedHashMap`**
   (b) `LinkedList`
   (c) `HashSet`
   (d) `Vector`
3. Wskaż metodę, która nie jest zdefiniowana w interfejsie `Collection`
   (a) **`sort`**
   (b) `toArray`
   (c) `add`
   (d) `isEmpty`

4. Która kolekcja wymaga aby implementacja metody `hashCode` była spójna z metoda `equals`?

   (a) **`HashMap`**
   (b) `CopyOnWriteArraySet`
   (c) `TreeMap`
   (d) `LinkedList`

5. Która metoda nie jest zdefiniowana w klasie `Collections`

   (a) **`isEmpty`**
   (b) `min`
   (c) `singleton`
   (d) `sort`

6. W jakiej kolejności przechowywane są obiekty w kolekcji `LinkedHashSet`

   (a) **w kolejności dodawania do kolekcji**
   (b) w kolejności rosnących wartości elementów
   (c) w kolejności przypadkowej
   (d) w kolejności definiowalnej przez użytkownika

7. Dlaczego mechanizm odwiedzania elementów w kolekcjach jest odseparowany od mechanizmu przechowywania kolekcji?

   (a) **ze względu na możliwość definiowania różnych strategii odwiedzania**
   (b) ze względu na konieczność używania typów parametryzowanych
   (c) ze względu na statyczną typizację
   (d) ze względu na wydajność