Rozdział 5.

1. **Klasa** – zbiór stanów oraz zachowań opisujący obiekty należące do tej samej kategorii; ma pola i metody. Definicja klasy jest realizowana za pomocą słowa kluczowego *class*.
2. **Diagramy UML** są wykorzystywane do projektowania klas, interfejsów itp. I zawierają metody, pola które później będą realizowane w klasach
3. **Obiekt** jest to instancja klasy, posiada własny zbiór pól, którego wartości określają jego indywidualność ; jest tworzony na podstawie klasy
4. **New** - w jezyku jawa służy do tworzenia nowych obiektów, po słowie New występuje nazwa odpowiedniego konstruktora, wartością zwracaną podczas tworzenia obiektu jest referencja do miejsca, gdzie obiekt utworzono
5. Różnica między metodą a konstruktorem: k. jest wywoływany w momencie tworzenia obiektu, jego nazwa musi być identyczna z nazwą klasy i nie może zwracać żadnej wartości. K. jest specyficzną metodą.
6. Laptop laptop firmowy = New coś tam
7. Składow klasy:
   pola – zmienne deklarowane we wnętrz klasy i zawierają informacje opisujące właściwości obiektów, jeśli nie ma jawnej inicjalizacji zmiennej to staje się ona domyślna
   metody – stanowią odpowiednik zachowań poszczególnych obiektów, podprogramy modelujące zachowania obiektów
8. Pola numeryczne – 0
   pola logiczne – false
   pola klasowe – null
9. Każda klasa posiada konstruktor, ponieważ jeśli nie został jawnie zadeklarowany to jest tworzony konstruktor domyślny
10. *Kod*
    -konstruktor nie może nic zwracać
11. *Kod*
    konstruktor ma być dwu parametrowy (nazwa, pojemność)
12. ***This*** oznacza referencje do bieżącego obiektu, czyli ułatwia dostęp do jego składowych, umożliwia wywołanie odpowiednich konstruktorów, jest wykorzystywany kiedy nazwa parametru metody jest taka sama jak nazwa pola.
13. ***Enkapsulacja*** – proces ukrywania składowych klas, uzywa się jej poprzez odpowiedniego modyfikatora dostępu (zwykle private)
14. *Kod* zacnie poprawny
15. **Pola statyczne** (klasowe) są oznaczone modyfikatorem static i są wspólne dla wszystkich obiektów danej klasy, dostęp do nich jest możliwy bez konieczności tworzenia obiektu klasy – pola niestatyczne (**instancyjne**)
16. Dostęp do metod statycznych jest możliwy bez konieczności tworzenia obiektu danej klasy, zatem metody te nie mogą odwoływać się do instancyjnych składowych klasy(lol) wiki: Metoda statyczna nie jest wywoływana na rzecz konkretnego obiektu, dlatego nie może odwoływać się do składowych niestatecznych.
17. Metody prywatne są wykorzystywane np. „Janusze nie mogą se zmieniać wynagrodzenia”- J.Tuchowski, 2015r. nie chcemy żeby użytkownik ingerował w nasze spox zmienne.

18. Metody dostępowe są używane do odczytu wartości pól (get i nazwa pola), a modyfikujące, zmiana pól (set i nazwa pola). Obie zazwyczaj nie zwracają żadnej wartości.
19. Kod
    null – taka wartość początkowa kurczaczki
20. *Kod*
    telefony[i] = new Telefon("!@#@#!"); trza dodać konstruktor

Rozdział 6.

ROZDZIAŁ 6

1. Czym jest kompozycja w programowaniu obiektowym?

Szczególny przypadek agregacji – obiekty składowe nie mogą istnieć bez obiektu głównego. Wyraża relację „składa się", „posiada". W skład tworzonej klasy może wchodzić dowolna liczba obiektów tworzonych na podstawie istniejących klas.

2. Wskaż różnice pomiędzy pojęciami klasa bazowa, a podklasa.

Podklasa dziedziczy a z bazowej dziedziczy podklasa.

3. Jakie składowe klasy bazowej dziedziczy podklasa?

Wszystkie, modyfikuje je i aby były bardziej wyspecjalizowane.

4. Czy każda klasa w Javie dziedziczy z innej klasy? Czy możliwe jest dziedziczenie z wielu klas?

Klasy które jawnie nie dziedziczą domyślnie dziedziczą z klasy Object.

5. Wyjaśnij pojęcie polimorfizmu.

Sposób wywołania metod – tworząc obiekty na podstawie klas dziedziczących można jako zmienną referencyjną podać typ klasy bazowej . Wywołując metody kompilator decyduje którą wykonać na podstawie typu obiektu na którą są wywoływane. [Możliwość traktowania obiektów wspólnego typu w ten sam sposób]

6. Jakie zadania spełnia klasa `Object`? Jakimi metodami dysponuje?

Główna klasa z której pośrednio lub bezpośrednio dziedziczą wszystkie klasy. Zawiera szereg metod zawierających operacje na obiektach. Najważniejsze z nich to:
clone() – tworzy kopię obiektu,
toString() – zwraca reprezentację obiektu w formie łańcucha tekstowego,
equals(Object) – porównuje dwa obiekty,
getClass() – zwraca nazwę klasy na podstawie, której powstał obiekt.

7. Podaj przykłady użycia metody `toString()` z klasy `Object`?

toString() – zwraca reprezentację obiektu w formie łańcucha tekstowego,

8. Jaką funkcję pełni operator `instanceof`?

służy do sprawdzenia czy dany obiekt należy do wskazanej klasy. Przyjmuje on wartość true lub false.

9. Wskaż różnice pomiędzy słowami kluczowymi `this` i `super`.

Super – dostęp do składowych klasy bazowej z klasy dziedziczącej / dostęp do konstruktora z klasy nadrzędnej

This – referencja do bieżącego obiektu dla ułatwienia dostępu do jego składowych jak i wywołanie odpowiednich konstruktorów.

10. Określ zalety używania pakietów.

Powiązane ze sobą klasy i interfejsy – szybsze odnalezienie właściwych; przejrzystość pisanych aplikacji; jasne określenie powiązań między klasami; unikanie konfliktów nazw; można korzystać z dodatkowych modyfikatorów dostępu

11. W jaki sposób możliwe jest użycie klas znajdujących się we wskazanym pakiecie?

Słowo package jako pierwsza instrukcja programu + nazwa pakietu.

12. Czy możliwe jest przesłonięcie metody `equals(Object)`? Jeśli tak, to jakie zadanie będzie spełniać ta metoda?
Equals jest przesłaniane w klasach pochodnych - porównywa obiekt który ją wywołuje oraz obiekt podany w parametrze metody

ROZDZIAŁ 7

1. Określ cel i zastosowanie interfejsów.

Zbiór wymagań dotyczących klas które będą go stosować – wyrażają sposób opisu funkcjonalności klasy bez określania jak to będzie uzyskane.

2. Czy specyfikacja języka Java narzuca ograniczenia na liczbę interfejsów, jakie może implementować tworzona klasa?

Może implementować dowolną liczbę - dziedziczenie wielobazowe

3. Czy możliwe jest tworzenie interfejsów pochodnych?

tak

4. W jakim przypadku klasa implementująca interfejs nie musi tworzyć wszystkich metod występujących w tym interfejsie?
Nie ma takiej sytuacji, zawsze musi

5. Które z poniższych deklaracji są błędne (`KlasaA`, `KlasaB`, `KlasaC` oznaczają nazwy klas, natomiast `InterfejsA`, `InterfejsB`, `InterfejsC` to nazwy interfejsów)?

```
class KlasaA extends KlasaB
class KlasaA implements InterfejsA
class KlasaC extends KlasaA, KlasaB
class KlasaB implements InterfejsB, InterfejsC
interface InterfejsA implements InterfejsB
interface InterfejsB extends InterfaceA
interface InterfejsC extends InterfaceA, InterfaceB
```

6. Odszukaj w dokumentacji, jakie interfejsy implementuje klasa `String`. Jakie nagłówki metod zdefiniowane zostały w tych interfejsach?

Serializable (writeObject, readObject, readObjectNoData), Comparable<String>(compareTo), CharSequence

7. Czy poprawny jest poniższy kod?

```
public interface Tester {
}
```

8. Podaj definicję klasy abstrakcyjnej.

To taka klasa która ma metody abstrakcyjne, nie mogą tworzyć obiektów, można na ich podstawie tworzyć klasy pochodne (które implementują wszystkie metody z klasy bazowej). Używana w procesie modelowania zawiera jedynie ogólne cechy i zachowania obiektów.

9. Wskaż różnice między klasą abstrakcyjną, a interfejsem.

Klasy abstrakcyjne mogą dziedziczyć tylko po jednej innej klasie, mogą dostarczać ciała metod i mogą określać metody o innej widoczności niż ‚public'.

Interfejsy mogą być dziedziczone wielokrotnie, mogą dostarczać domyślne ciała metod (ale nie mogą przesłaniać innych metod) i dotyczą tylko metod o widoczności public.

10. Poniższy kod programu zawiera definicję klasy abstrakcyjnej wraz z jej składowymi. Wskaż ewentualne błędy.

Jeżeli metoda w klasie abstract ma identyfikator abstract to musi składać się wyłącznie z deklaracji nazwy.

```
public abstract class Zwierze {
     private String nazwa;
     public abstract void je() {
     System.out.println ("Aktualnie je.");
     }
     public abstract void idzie();
     public abstract void pije();
}
```

11. Czy możliwe jest utworzenie obiektów na bazie klasy abstrakcyjnej?

Nie

12. Czy nazwa klasy, która dziedziczy z klasy abstrakcyjnej i nie dostarcza implementacji metod zawartych w nadklasie musi zostać poprzedzona identyfikatorem `abstract`?

---

13. Wskaż błąd w poniższym kodzie programu.

Nie można dziedziczyć z klas finalnych

```
final class A {
     private String nazwa;
     public String pobierzNazwe() {
     return this.nazwa;
     }
}
class B extends A {
```

```
    private int liczba;
    public int pobierzLiczbe() {
    return liczba;
    }
}
```

14. Jakie własności posiadają metody finalne?

Nie jest możliwe ich przesłonięcie w klasach dziedziczących.

15. Czym są klasy anonimowe? Wskaż ich zastosowanie.

Klasy wewnętrzne nie posiadające nazwy, mają ograniczone użycie, jej definicja znajduje się w ciele innej klasy, posiada dostęp do wszystkich składowych klasy w której została zdefiniowana (też składowych prywatnych); bez ograniczeń wykorzystania dziedziczenia i implementacji interfejsów.

1. **Kompozycja** – wyraża relacje „składa się z" lub „posiada" . w skład tworzonej klasy może wchodzić dowolna liczba obiektów tworzona na podstawie istniejących klas.
2. Różnica pomiędzy pojęciami klasa bazowa (ta z której dokonano dziedziczenia) a podklasa (jest to też dziecko o_O; ta która dziedziczy).
3. Podklasa dziedziczy składowe klasy bazowej - wszelkie cechy i zachowania(pola i metody), dodając bądź modyfikując je by były bardziej wyspecjalizowane, konstruktory **nie** są dziedziczone.
4. Klasy które nie posiadają zadeklarowanego dziedziczenia, domyślnie dziedziczą z klasy Object, nie jest możliwe dziedziczenie z wielu klas.
5. **Polimorfizm** –Możliwość traktowania obiektów różnych podtypów pewnego wspólnego typu, w ten sam sposobik . inaczej wielopostaciowość poleg na tym, że do ogólnej wspólnej referencji można przypisać obiekty różnych typów, które po typie tej referencji dziedziczą.
6. **Klasa Object** jest główną klasą z której pośrednio lub bezpośrednio dziedziczą wszystkie klasy. Zawiera szereg metod zawierających operacje na obiektach. Najważniejsze z nich to:
   clone() – tworzy kopię obiektu,
   toString() – zwraca reprezentację obiektu w formie łańcucha tekstowego,
   equals(Object) – porównuje dwa obiekty,
   getClass() – zwraca nazwę klasy na podstawie, której powstał obiekt.
7. Przykłady użycia metody toString() z klasy Object –jak wyżej ziom
8. Operator instanceof służy do sprawdzenia czy dany obiekt należy do wskazanej klasy. Przyjmuje on wartość true lub false.
9. Różnice pomiędzy słowami kluczowymi this i super:
   this znajdujaca sie w konstruktorze, wywołuje inny konstruktor z tej samej klasy
   super - wywołuje konstruktor z klasy nadrzędnej, zapewnia także dostęp do składowych klasy bazowej z klasy dziedziczącej
10. Zalety pakietów –pozwala na szybsze odnalezienie klas i interfejsów, wpływa korzystnie na przejrzystość pisanych aplisracji, jasne określenie związków między klasami, unikanie konfliktów nazw, możliwość z skorzystania dodatkowych modyfikatorów dostępu

11. możliwe jest użycie klas znajdujących się we wskazanym pakiecie – umożliwia kluczowe słowo *import* a potem nazwa pakietu oraz nazwa klassy albo * na wszyyyyystkie klasy należące do pakietu
12. Jest możliwe przesłonięcie metody equals, będzie porównywa obiekt który ją wywołuje oraz obiekt podany w parametrze metody

Rozdział 7.

1. Cel i zastosowanie interfejsu – wymusza implementacje metod, które są w nim zawarte prze klasy które go implementują.
2. Można implementować wiele interfejsów
3. Dopuszczalne jest dziedziczenie interfejsów
4. Nie musi implementować metody jak już ją kurde posiada
5. Class nie może extends kilku klas (dziedziczenie z 1)
   interfejs nie może implementować interfejsu (ale dziedziczyć może i to z wielu np. interface InterfejsC extends InterfaceA, InterfaceB)
6.
7. Może być interfejs bez ciała
8. Klasa abstrakcyjne – wykorzystywane w procesie modelowania zawierając jedynie ogólne cechy, deklaracja odbywa się zza pomocą słowa kluczowego abstrakt, nie może stanowić podstawy do utworzenia obiektu, na jej podstawie możliwe jest tworzenie klas pochodnych które muszą zaimplementować wszystkie metody abstrakcyjne występujące w klasie bazowej.
9. Przede wszystkim w klasie abstrakcyjnej możemy już umieścić jakąś implementację, natomiast klasy rozszerzające będą miały jedynie obowiązek rozbudować funkcjonalność poprzez zaimplementowanie metod abstrakcyjnych. Klasy abstrakcyjne wymuszają w klasach pochodnych implementacje wszystkich metod abstrakcyjnych.
10. Metoda abstrakcyjna nie może mieć ciała huehue duszek kurwa Kacperek
11. Nie można tworzyć na jej bazie obiektu (tylko dziedziczenie)
12. Klasa która dziedziczy z klasy abstrakcyjnej i nie dostarcza implementacji metod nadklasy musi być poprzedzona identyfikatorem *abstact.* (nie jest abstrakt jak już ma ciało metody abstrakcyjnej)
13. Nie można dziedziczyć z klas finalnych *final*
14. Metody finalne – nie możliwe jest ich przesłonięcie w klasach dziedziczących, czyli nie da się z nich dziedziczyć
15. Klasy anonimowe to klasy wewnętrzne, nie posiadające nazwy, wykorzystywane do obsługi zdarzeń w programowaniu aplikacji korzystających z GUI SRUI
16. Jakie są różnice pom. metodą, a konstruktorem – konstruktor jest specyficzną metodą, wywoływany zawsze podczas tworzenia obiektu, metoda niekoniecznie. Konstruktor musi nazywać się jak klasa i nic nie zwraca, w jego skład wchodzą instrukcje, które zostaną wykonane podczas tworzenia obiektu, można je przeciążać podobnie jak metodę
17. Czy kazda klasa dziedziczy z innej klasy? Klasa Object nie posiada klasy bazowej
18. Czym jest blok instrukcji i jaki jest sposób notacji w języku programowania? jest to miejsce zbiorczego przetwarzania instrukcji ujęte w nawiasach klamrowych
19. Jaki typ danych może przyjmować wyrażenie instrukcji switch: byte, short, int, long, char

20. Klauzula default nie jest każdorazowo wymagana, jest używana gdy nie został użyty żaden z case
21. Jakie składowe wchodzą w skład instrukcji iteracyjnej for? wyr początkowe, wyr logiczne, wyr modyfikujące
22. Czym charakteryzuje się odmiana instrukcji for przeznaczona do operacji na elementach tablicy? Liczba iteracji to liczba elementów tablicy
23. Jakie zasadnicze różnice cechują pętlę określoną for oraz pętlę nieokr. while? Znana/ nieznana liczba iteracji
24. Czym różnią się instrukcje iteracyjne while oraz do… while? While nieokreślona, działa wd. warunku logicznego, wykonuje dopóki wyr. nie przyjmuje wart fałsz, do while w przeciwieństwie do while wykona blok instrukcji przynajmniej raz
25. W jaki sposób utworzone obiekty usuwane sa z pamięci operacyjnej? Obiekty istnieją tak długo jak odwołania do nich – java to okresowo sprawdza i usuwa automatycznie przez garbage collector
26. Jakie istotne cechy charakteryzują metodę instancyjną oraz metodę klasową?
    Instancyjna nie ma static, działa tylko na obiekty danej klasy, klasowa ma modyfikator static i nie trzeba tworzyć obiektu.
27. Która instrukcja kończy działanie metody? break, ew. continue
28. Dlaczego metody klasy Math zostały zdefiniowane jako klasowe? Bo można je wywołać nie tworząc obiektu
29. Które z klas należą do kat. klas opakowujących? Long, Int, String, Boolean, Char