1. Skladanie - vytvorenie inštancie existujúcej triedy v novej triede
2. Agregácia - objekt jednej triedy obsahuje objekt inej triedy (vzťah “má”)

```
13 public class Obchodny_dom {
14
15     static public int vytahpocet;
16     static public int staznosti;
17     public int zarobok;
18     public int cistota = 5;
19     public int energia = 250;
20     public int pocet_vybavenych, pocet_vyhodenych;
21     public int pocet_robotov;
22
23     Vytah Vyta;
24     Sklad Skladik;
25     Dom_jedlo PrvaCast;
26     Dom_ostatne DruhaCast;
27
28     Obchodny_dom() {
29         this.Vyta = new Vytah();
30         this.Skladik = new Sklad();
31         this.PrvaCast = new Dom_jedlo();
32         this.DruhaCast = new Dom_ostatne();
33     }
34
35     public static void main(String[] args) {
36         Scanner sc = new Scanner(System.in);
37         Random generator = new Random();
38
39         Obchodny_dom Dom = new Obchodny_dom();
40         Zakaznik[] Zak = new Zakaznik[250];
41         Robot[] Rob = new Robot[5];
42         Pokladnik[] Pok = new Pokladnik[5];
```

3. Kompozícia - agregujúci objekt nesie zodpovednosť za existenciu a uloženie agregovaných objektov (vzťah “vlastní”)
4. Zapúzdrenie - zaistenie, aby s dátami nebolo možné z vonkajšku triedy manipulovať inak, než pomocou metódy tejto triedy.

```
28     Obchodny_dom() {
29         this.Vyta = new Vytah();
30         this.Skladik = new Sklad();
31         this.PrvaCast = new Dom_jedlo();
32         this.DruhaCast = new Dom_ostatne();
33     }
34
35     public static void main(String[] args) {
36         Scanner sc = new Scanner(System.in);
37         Random generator = new Random();
38
39         Obchodny_dom Dom = new Obchodny_dom();
```

vlastní Vytah, Sklad, Dom_jedlo, Dom_ostatne

5. Dedenie - prevzatie formy existujúcej triedy ktorú je možné rozšíriť

```java
public class Zamestnanec extends INTERACTION_OBJECT {
    public String Meno_Priezvisko;

    Zamestnanec(int poz, String Meno) {
        super(poz);
        this.Meno_Priezvisko = Meno;
    }

    public static void alarm(Zamestnanec zam) {
        System.out.println("!!!ALARM BOL ZAPNUTY!!!");
        System.out.println("!!!OBCHOD JE EVAKUOVANY!!!");
    }
}
```

rozširuje o atribút int rychlost

```java
import manipulate_objects.Pokladna;

public class Pokladnik extends Zamestnanec {
    int rychlost;

    public Pokladnik(int fast, String meno, Pokladna stroj) {
        super(3, meno);
        this.rychlost = fast;
        stroj.pouzivana = true;
    }
}
```

6.a) preťaženie - rovnomenné metódy líšiace sa parametrami
b) prekonanie - rovnomenné metódy ktoré sa v podtype nelíšia argumentami. Metóda v podtype prekryje nadtypovú.

a)

```java
public class Robot extends INTERACTION_OBJECT {
    boolean cisti = true;
    public boolean tovar = false;
    public boolean security = false;

    public Robot(Obchodny_dom Dom) {
        super(3);
        Dom.energia -= 20;
        Dom.pocet_robotov++;
    }

    public Robot(boolean tova, Obchodny_dom Dom) {
        super(3);
        Dom.energia -= 30;
        Dom.pocet_robotov++;
        this.tovar = tova;
    }

    public Robot(boolean tova, boolean secu, Obchodny_dom Dom) {
        super(3);
        Dom.energia -= 50;
        Dom.pocet_robotov++;
        this.tovar = tova;
        this.security = secu;
    }

    static public void cisti(Robot cist, Obchodny_dom Dom) {
        Dom.cistota = 5;
        Dom.energia -= 2;
    }
```

b)

```java
public class Zamestnanec extends INTERACTION_OBJECT {
    public String Meno_Priezvisko;

    Zamestnanec(int poz, String Meno) {
        super(poz);
        this.Meno_Priezvisko = Meno;
    }

    public static void alarm(Zamestnanec zam) {
        System.out.println("!!!ALARM BOL ZAPNUTY!!!");
        System.out.println("!!!OBCHOD JE EVAKUOVANY!!!");
    }
}
```

```java
import systems.Vytah;

public class Administrator extends Zamestnanec {

    public Administrator(int cash, String meno) {
        super(3, meno);
    }

    public static void oprava(Administrator admin) {
        Vytah.funkcnost = true;
    }

    public static void alarm(Zamestnanec admin) {
        System.out.println("!!!ALARM BOL ZAPNUTY!!!");
        System.out.println("!!!OBCHOD JE EVAKUOVANY!!!");
    }
}
```

Rozšírenie ďalších objektovo-orientovaných princípov v 3. zadaní

7. Abstraktné triedy - definovanie spoločných rysov podtried (nieje z nich možné vytvoriť inštanciu)

```java
package interaction_objects;

abstract public class Clovek {
    public int pozicia;

    Clovek(int i) {
        this.pozicia = i;
    }
}
```

8. Rozhrania

```java
package interaction_objects;

public class Zamestnanec extends Clovek implements Pracujuci {
    public String Meno_Priezvisko;

    Zamestnanec(int poz, String Meno) {
        super(poz);
        this.Meno_Priezvisko = Meno;
    }

    public void alarm() {
        System.out.println("Obycajny zamestnanec v panike nevie najst zapnutie alarmu");
    }
}
```

```java
package interaction_objects;

public interface Pracujuci {

    public default void vyplatit()  {
        System.out.println("Pracujuci clovek bol vyplateny");
    }
}
```

9. Uppercasting a polymorfizmus

```
// ALARM
if (Zak[i].spokojnost == 2)
    poplach(Pok[0]);
else if (Zak[i].spokojnost < 2)
    poplach(Admin2);
```

kedže je tam if else, bolo by možné urobiť to aj bez polymorfizmu, je to však ukážka polym. cez poplach, ktorý tu zobere ako argument 2 rôzne triedy

```
60          Zamestnanec Ad = new Administrator("Marek Horvath");
61          Administrator Admin1 = (Administrator) Ad;
62          Zamestnanec Ad2 = new Administrator("Jana Figlova");
63          Administrator Admin2 = (Administrator) Ad2;
```

pretypovanie na podtyp

```
5  public class Pokladnik extends Zamestnanec {
6      int rychlost;
7
8      public Pokladnik(int fast, String meno, Pokladna stroj) {
9          super(3, meno);
10         this.rychlost = fast;
11         stroj.pouzivana = true;
12     }
13
14     public void alarm() {
15         System.out.println("!!!ALARM BOL ZAPNUTY!!!");
16     }
17
18 }
19
```

```
5  public class Administrator extends Zamestnanec {
6
7          public Administrator(String meno) {
8              super(3, meno);
9          }
10
11         public static void oprava(Administrator admin) {
12             Vytah.funkcnost = true;
13         }
14
15         public void alarm() {
16             System.out.println("!!!ALARM BOL ZAPNUTY!!!");
17             System.out.println("!!!ADMINISTATOR EVAKUUJE OBCHOD!!!");
18         }
19 }
```

```
14 public class Obchodny_dom {
15
16     static public int vytahpocet;
17     static public int staznosti;
18     public int zarobok;
19     public int cistota = 5;
20     public int energia = 250;
21     public int pocet_vybavenych, pocet_vyhodenych;
22     public int pocet_robotov;
23
24     Vytah Vyta;
25     Sklad Skladik;
26     Dom_jedlo PrvaCast;
27     Dom_ostatne DruhaCast;
28
29     Obchodny_dom() {
30         this.Vyta = new Vytah();
31         this.Skladik = new Sklad();
32         this.PrvaCast = new Dom_jedlo();
33         this.DruhaCast = new Dom_ostatne();
34     }
35
36     private static void poplach(Zamestnanec zam) {
37         zam.alarm();
38     }
```