Part 1

- Fun with collections again (lookup the documentation id needed)
- Imagine you have collection of arbitrary objects:
  ```
  val l = List(1, "hello", 2.56, 0x45, “key")
  ```
- We want to write a helper function that would allow to represent it in textual form (i.e. type and value):
  ```
  val s = stream(l)
  println(s)
  // result Int(1) String(hello) Double(2.56) Int(69)
  String(key)
  ```
- Write an implementation of such function using collection and patterns matching

Part 2

- The idea is to write an extractor helpful in decoding an address. Example:
```
val Address(str, bld, ap) = "Krakowska street 34/7"
println(“str:”+str +" bld:”+bld+" ap:“+ap)
//—> this gives: str:Krakowska bld:34 ap:7
val Address(str2, bld2, ap2) = "River street 22” // this should also wor
//  val Address(ul3, dom3, m3) = "Reymont" // here we want to fail (i.e.
extractor returning None
```

- Once this is debugged and working we expand this example as follows. We want the extractor: AddressX that is able to extract two these two kinds of addresses packaged in the case classes AddressInfo1 (this one has street name and number) and AddressInfo2 (this one has street name, building number and apartment number). The extraction would look more or less like this:
```
val AddressX(a1) = "Krakowska street 34/7” // a1 is AddressInfo2
val AddressX(a2) = "Krakowska street 17” // a2 is AddressInfo1
```

- Place couple of such addresses in the List and using expression matching & collections operations write helper function filtering all addresses that are addresses of apartments on a given street (i.e. are objects of type AddressInfo1)
```
val ap = filterApparments(myAddresses, “Krakowska”)
```