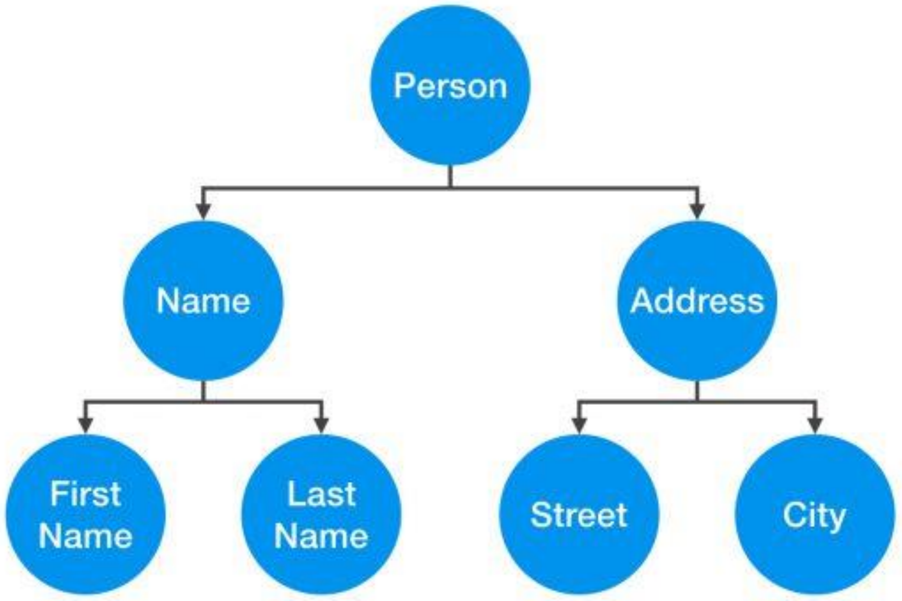
CS F213 - Object Oriented Programming

J. Jennifer Ranjani
email: jennifer.ranjani@pilani.bits-pilani.ac.in
Chamber: 6121 P, NAB
Consultation: Appointment by e-mail

https://github.com/JenniferRanjani/Object-Oriented-Programming-with-Java

BITS Pilani
Pilani Campus

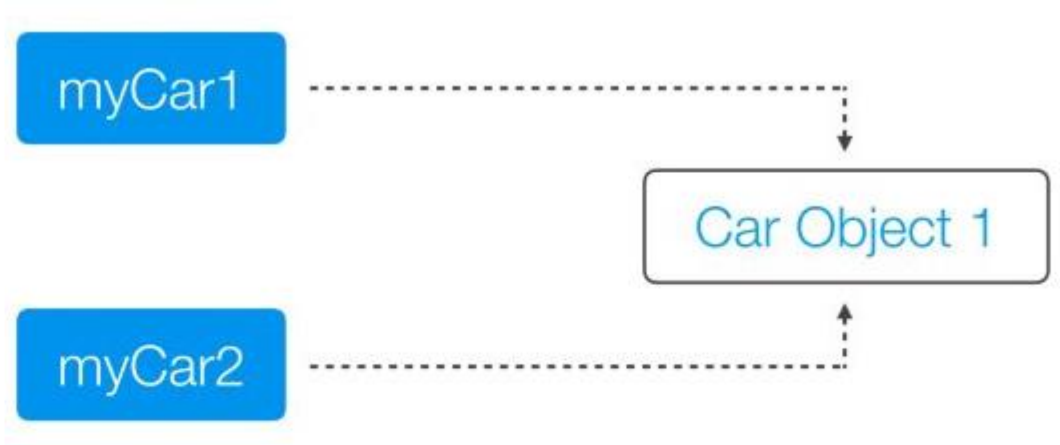
Objects – an Example



Person is made up of Name and Address objects which in turn is made up of objects like FirstName, LastName, Street and City respectively

Copying Objects

- When we use assignment operator it will create a copy of reference variable and not the object.



- Cloning refers to creation of exact copy of an object

- It creates a new instance of the class of current object and initializes all its fields with exactly same contents.

Cloning Condition

x.clone() != x

x.clone().equals(x) return true

x.clone().getClass() == x.getClass()

“ clone should be a new object but it should be equals to its original”

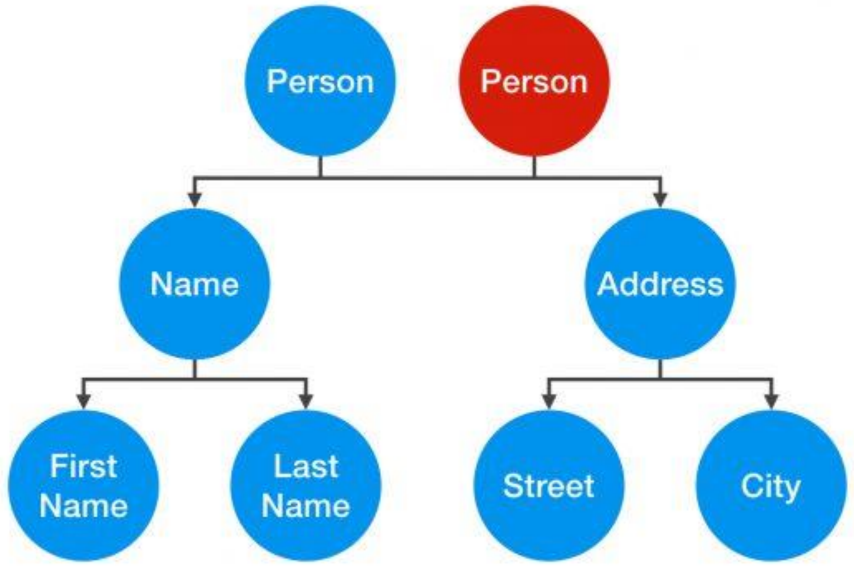
Clone requirements

Any class willing to be cloned must

1. **Declare the clone() method to be public**
2. **Implement Cloneable interface**

```
class Account implements Cloneable
{
public Object clone()
{
 try { super.clone() }
catch(CloneNotSupportedException e){ .. }
}
```

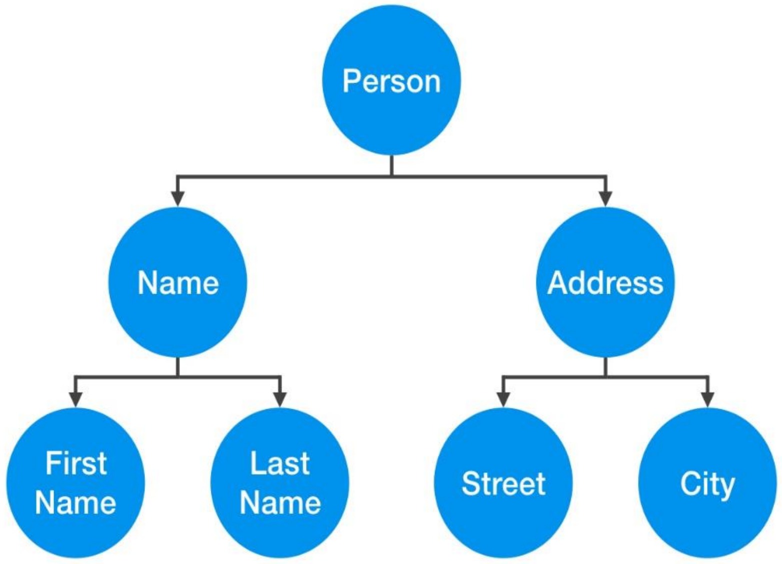
Shallow Copy

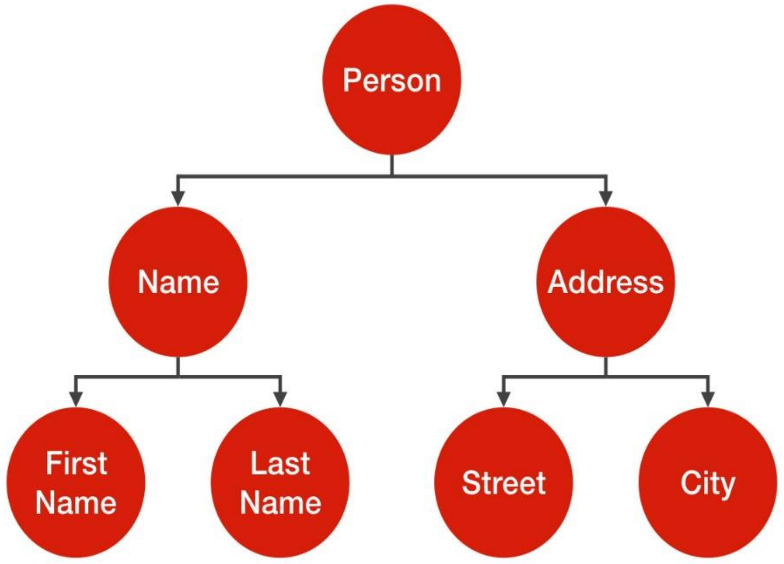


- It copies the main object but doesn’t copy the inner objects.
- Inner objects are still shared between the original and its copy

Deep Copy

- It is a fully independent copy of an object and it copies the entire object structure





Java Type System

- Type: Set of values with a set of operations that can be applied to the values

- Example:
  - int type
  - Account type

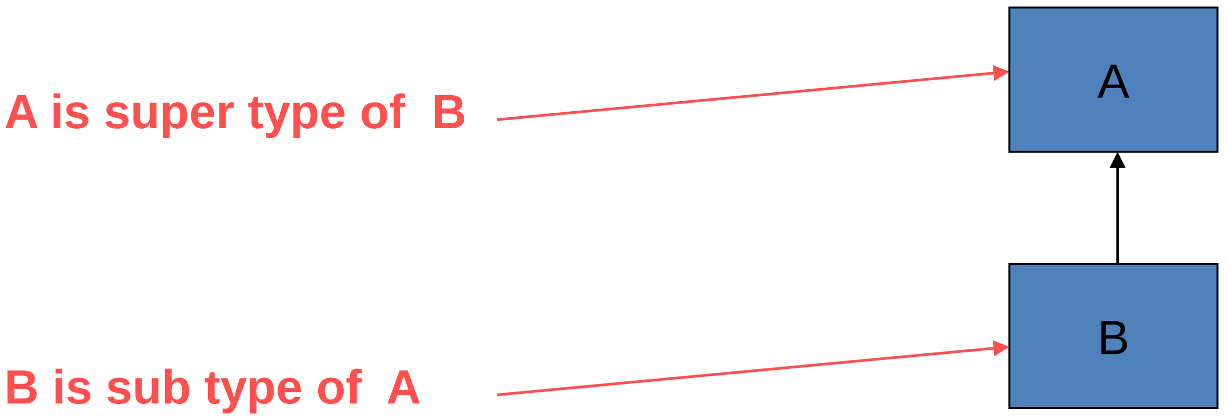
- Java is a strongly typed language
  - Compile-time check
    ```
    Employee e = new Employee();
    e.clear(); // ERROR
    ```
  - Run-time check:
    ```
    e = null;
    e.setSalary(20000); // ERROR
    ```

Types in Java

- Primitive types:
  - int short long byte
  - char float double boolean
- Class types
- Interface types
- Array types
- The null type
- Note: void is not a type

Sub types

- Sub type specifies the inheritance relationship either by extending a class or implementing an interface



Example

What's Expected

$X \quad x1 \quad = \quad ?$

- If X is an interface
  - RHS can be an instance of any class implementing X.
- If X is abstract class
  - RHS can be an instance of any concrete subclass of X
- If X is a concrete class
  - RHS can be either an instance of X or any of its subclasses

Rules for Subtype Relationship

S is a subtype of T if

1. S and T are the same type
2. S and T are both class types, and T is a direct or indirect superclass of S
3. S is a class type, T is an interface type, and S or one of its superclasses implements T
4. S and T are both interface type, and T is a direct or indirect superinterface of S
5. S and T are both array type, and the component type of S is a subtype of the component type of T
6. S is not a primitive type and T is the type Object
7. S is an array type and T is Cloneable or Serializable
8. S is the null type and T is not a primitive type

Review Questions

1. Is Container is a subtype of Component ?.
2. Is JButton is a subtype of Component ?
3. Is FlowLayout is a subtype of LayoutManager?
4. Is ListIterator is a subtype of Iterator ?
5. Is Rectangle[ ] is a subtype of Shape[ ] ?
6. Is int[ ] is a subtype of Object ?
7. Is int is subtype of long ?
8. Is long is a subtype of int ?
9. Is int is a subtype of Object ?