Java Programming

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

Today's Lecture

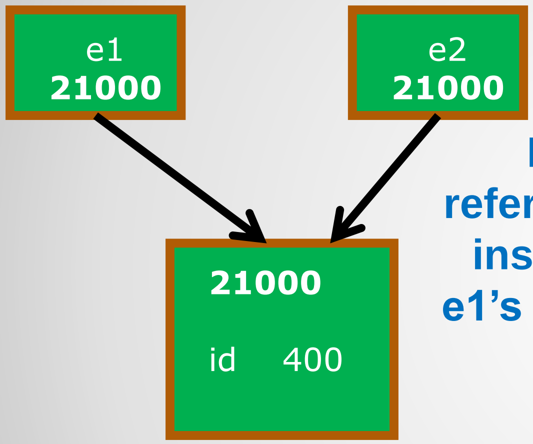
- Shallow Copy
- Deep Copy

```
public class Employee {
   public int id;
}
public static void main(String[] args)
{
   Employee e1 = new Employee();
   e1.id = 400;
   Employee e2 = e1;
}
```

Shallow Copy

```
Employee e1 = new Employee();
e1.id = 400;
Employee e2 = e1;
```

Shallow copy (just copies the address)



Both e1 and e2 are referring to the same exact instance of Employee. If e1's id is changed it is also changing e2's id.

Shallow Copy

- Now a simple deep copy example…

Simple Deep Copy Example

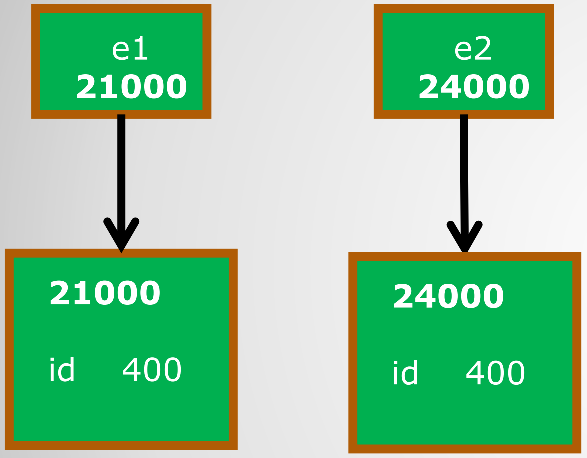
```
public class Employee {
   public int id;
}
public static void main(String[] args)
{
   Employee e1 = new Employee();
   e1.id = 400;
   Employee e2 = new Employee();
   e2.id = e1.id;
}
```

Deep Copy

```
Employee e1 = new Employee();
e1.id = 400;
Employee e2 = new Employee();
e2.id = e1.id;
```

Create a new instance of Employee

Copy the id from e1 to e2



If e1's id is changed it will not effect e2's id

Deep Copy

Shallow Copy with One Class Inside of Another

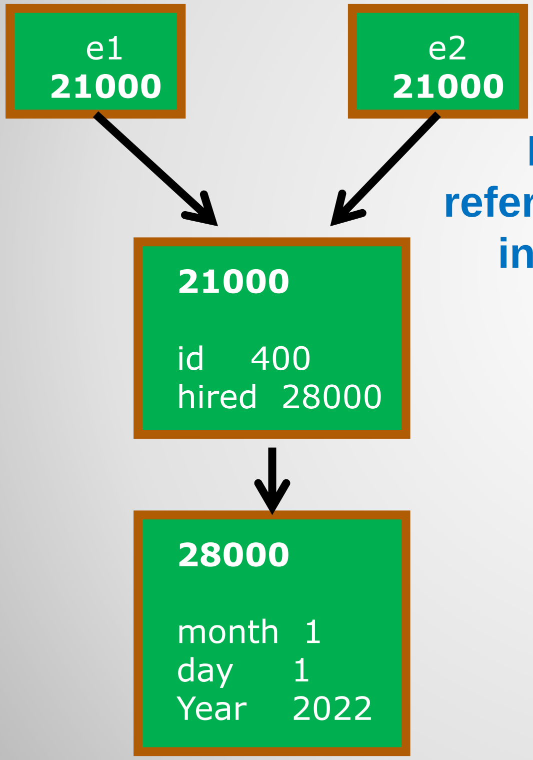
- Shallow copy with one class inside of another example…

Shallow Copy

```
public class Date {
  public Date(int m, int d, int y) { month=m;day=d;year=y;}
  public int year;
  public int month;
  public int day;
}
public class Employee {
  public int id;
  public Date hired;
}
public static void main(String[] args)
{
  Employee e1 = new Employee();
  e1.hired = new Date(1, 1, 2022);
  Employee e2 = e1;
}
```

```
Employee e1 = new Employee();
e1.hired = new Date(1, 1, 2022);
Employee e2 = e1;
```

Shallow copy (just copies the address)



Both e1 and e2 are referring to the same exact instance of Employee

Shallow Copy

Better But Not a Full Deep Copy

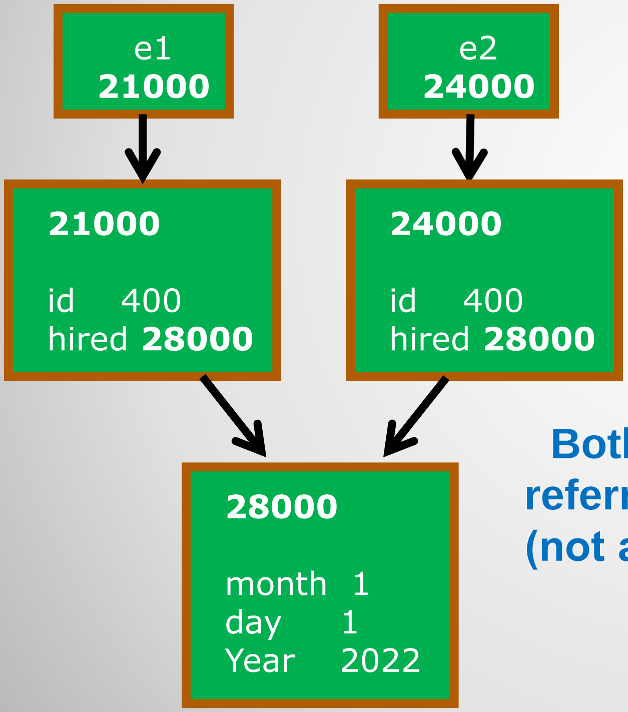
- The next example tries to do a deep copy but does not do it fully…

```
public class Date {
   public Date(int m, int d, int y) { month=m;day=d;year=y;}
   public int year;
   public int month;
   public int day;
}
public class Employee {
   public int id;
   public Date hired;
}
public static void main(String[] args) {
   Employee e1 = new Employee();
   e1.hired = new Date(1, 1, 2022);
   Employee e2 = new Employee();
   e2.id = e1.id;
   e2.hired = e1.hired;
}
```

Better But Not a Full Deep Copy

```
Employee e1 = new Employee();
e1.hired = new Date(1, 1, 2022);
Employee e2 = new Employee();
e2.id = e1.id;
e2.hired = e1.hired;
```

Shallow copy of member variable (just copies the address). Need to update so that a new instance of Date is created for e2.



Both e1 and e2 have their hired members referring to the same exact instance of Date (not a deep copy). If e1's hired is changed it will also be changing e2's hired.

Better But Not a Full Deep Copy

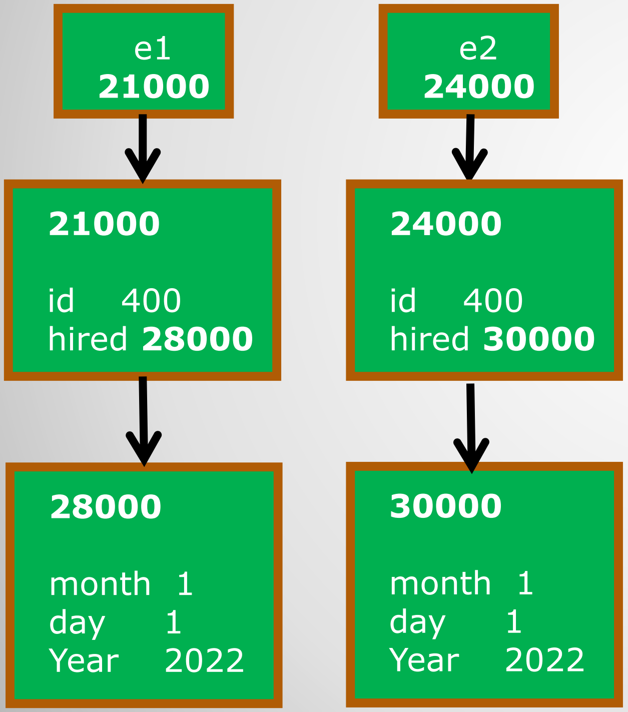
```
public class Date {
   public Date(int m, int d, int y) { month=m;day=d;year=y;}
   public int year;
   public int month;
   public int day;
}
public class Employee {
   public int id;
   public Date hired;
}
public static void main(String[] args) {
   Employee e1 = new Employee();
   e1.hired = new Date(1, 1, 2022);
   Employee e2 = new Employee();
   e2.id = e1.id
   e2.hired = new Date(e1.hired.month, e1.hired.day, e1.hired.year);
}
```

Deep Copy

```
Employee e1 = new Employee();
e1.hired = new Date(1, 1, 2022);
Employee e2 = new Employee();
e2.id = e1.id
```

e2.hired = new Date(e1.hired.month, e1.hired.day, e1.hired.year);

Make a new instance of Date for e2 and copy the month, day, and year from e1



e2 is a deep copy of e1. If anything is changed in e1 it will not effect e2

Deep Copy

- **End of Slides**

End of Slides