Object References

How Java's Memory Handles Objects

Alwin Tareen

Object References

- Consider the following class, which models a pizza:

```
public class Pizza
{
    private String topping;
    private double price;

    public Pizza(String t, double p)
    {
        topping = t;
        price = p;
    }
```

Object References

```
    public String getTopping()
    {
        return topping;
    }

    public double getPrice()
    {
        return price;
    }

    public void setPrice(double p)
    {
        price = p;
    }
}
```

Object References

```
public class PizzaTest
{
    public static void main(String[] args)
    {
        Pizza pyro = new Pizza("Cheese", 9.25);
        Pizza dominos = null;
        dominos = pyro;
        dominos.setPrice(12.50);
        double cost = pyro.getPrice();
        System.out.println(cost);
    }
}
```

Object References

Pointing to a Memory Address

- Consider what happens in `Java`'s memory when the object reference `pyro` is created.
- `pyro` is automatically assigned a six-digit hexadecimal number. This number corresponds to a specific location in `Java`'s memory banks. The object reference `pyro` points to that location.

`pyro`		memory address
`2A4C97`	$\rightarrow$	`2A4C97`

- In other words, `pyro` only contains a memory location address, not the object itself.

Object References

Allocating memory for the object

- The entire `Pizza` object resides at the memory location that is specified by `pyro`.
- This object allocates a sufficient amount of memory to hold both the `topping` and the `price` instance variables, as well as the methods `getTopping()`, `getPrice()`, and `setPrice()`.

Setting an object reference to `null`

- Note that you can create an object reference without placing a memory location inside of it. Simply set the object reference to `null`.

```
Pizza dominos = null;
```

Aliasing

Object references pointing to the same address

- Consider the case where I copied the memory location from `pyro` into `dominos`.
- Then, the two object references would be pointing to the same address location in memory. This is called **aliasing**.

Altering the `price`

- Note how we can run the `setPrice()` method using the `dominos` object reference. We can alter the `price` of the pizza in this manner.

```
dominos.setPrice(12.50);
```

Object References: End of Notes