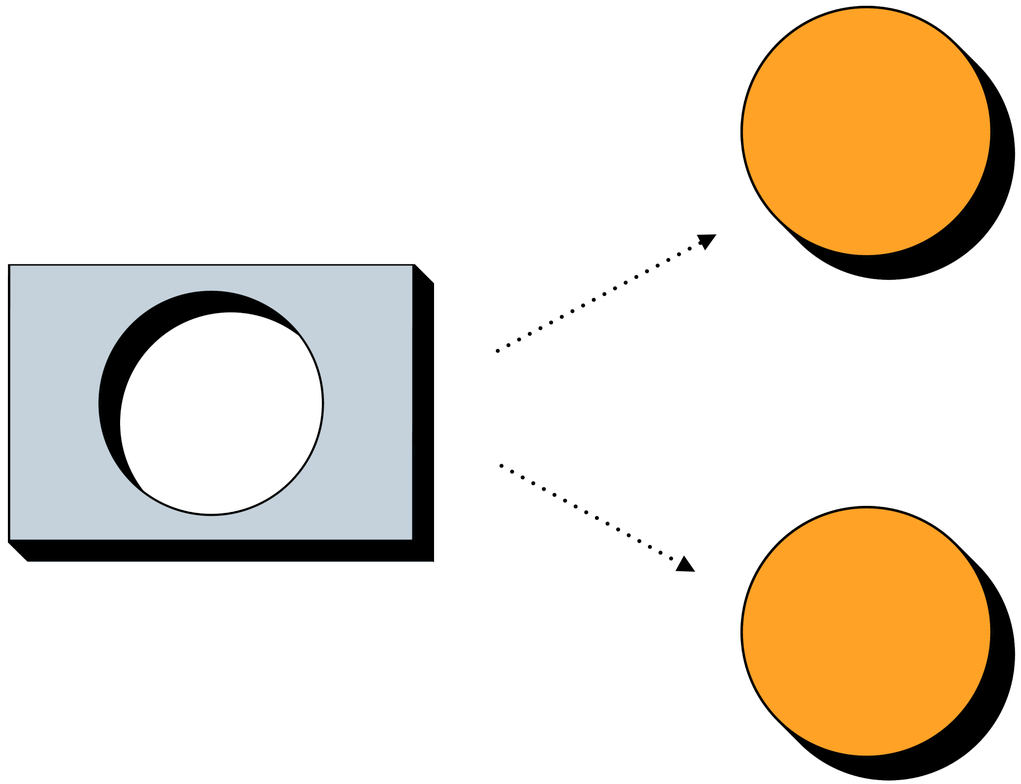
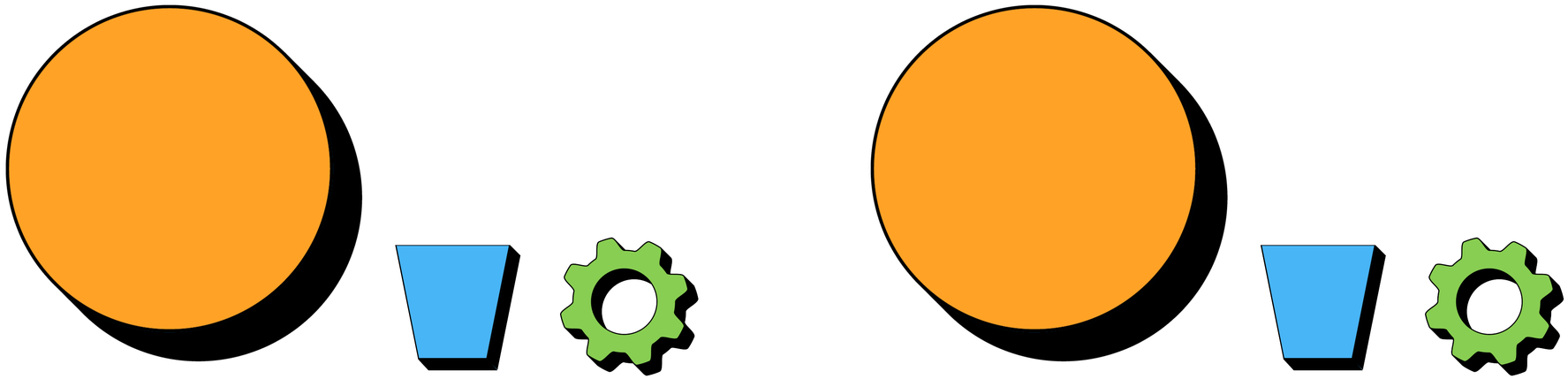
Essential Computing 1

Classes & Objects



Objects contain variables (attributes) and methods



You have already worked with objects

variable of type Scanner

```
Scanner scanner = new Scanner( system.in );

System.out.println( "You typed: " + scanner.nextLine() );
```

You have already worked with objects

variable of type Scanner

object of type Scanner being "instantiated" (created)

```
Scanner scanner = new Scanner( system.in );

System.out.println( "You typed: " + scanner.nextLine() );
```

You have already worked with objects

variable of type Scanner

object of type Scanner being "instantiated" (created)

```
Scanner scanner = new Scanner( system.in );

System.out.println( "You typed: " + scanner.nextLine() );
```

method inside the scanner object

Defining a class for instantiating monster objects

```
class Monster
{
    int health;
    int strength;

    void hit( Monster other )
    {
        other.health -= (int) (Math.random() * strength );
    }
}
```

Class names start with capital letter

```
class Monster
{
    int health;
    int strength;

    void hit( Monster other )
    {
        other.health -= (int) (Math.random() * strength );
    }
}
```

Class scope

```
class Monster
{
    int health;
    int strength;

    void hit( Monster other )
    {
        other.health -= (int) (Math.random() * strength );
    }
}
```

Variables that belong to the object (when it is instantiated)

```
class Monster
{
    int health;
    int strength;

    void hit( Monster other )
    {
        other.health -= (int) (Math.random() * strength );
    }
}
```

Method that belong to the object (when it is instantiated)

```
class Monster
{
    int health;
    int strength;

    void hit( Monster other )
    {
        other.health -= (int) (Math.random() * strength );
    }
}
```

Here we are receiving another monster as argument

```
class Monster
{
    int health;
    int strength;

    void hit( Monster other )
    {
        other.health -= (int) (Math.random() * strength );
    }
}
```

Here we are hitting the other monster, giving damage.

```
class Monster
{
    int health;
    int strength;

    void hit( Monster other )
    {
        other.health -= (int) (Math.random() * strength );
    }
}
```

Testing the monster class in the main method

```
public class Main {
    public static void main( String[] args )
    {
        Monster cyclops = new Monster();
        cyclops.health = 100;
        cyclops.strength = 10;

        Monster dragon = new Monster();
        dragon.health = 100;
        dragon.strength = 30;

        cyclops.hit( dragon );

        System.out.println( "dragon health: " + dragon.health );
    }
}
```

Instantiating two monster objects

```
public class Main {
    public static void main( String[] args )
    {
        Monster cyclops = new Monster();
        cyclops.health = 100;
        cyclops.strength = 10;

        Monster dragon = new Monster();
        dragon.health = 100;
        dragon.strength = 30;

        cyclops.hit( dragon );

        System.out.println( "dragon health: " + dragon.health );
    }
}
```

Setting their variables (accessed using dot syntax)

```
public class Main {
    public static void main( String[] args )
    {
        Monster cyclops = new Monster();
        cyclops.health = 100;
        cyclops.strength = 10;

        Monster dragon = new Monster();
        dragon.health = 100;
        dragon.strength = 30;

        cyclops.hit( dragon );

        System.out.println( "dragon health: " + dragon.health );
    }
}
```

Calling a method in the first monster object,
passing the second monster object as argument.

```
public class Main {
    public static void main( String[] args )
    {
        Monster cyclops = new Monster();
        cyclops.health = 100;
        cyclops.strength = 10;

        Monster dragon = new Monster();
        dragon.health = 100;
        dragon.strength = 30;

        cyclops.hit( dragon );

        System.out.println( "dragon health: " + dragon.health );
    }
}
```

Getting the health variable from the second monster object

```
public class Main {
    public static void main( String[] args )
    {
        Monster cyclops = new Monster();
        cyclops.health = 100;
        cyclops.strength = 10;

        Monster dragon = new Monster();
        dragon.health = 100;
        dragon.strength = 30;

        cyclops.hit( dragon );

        System.out.println( "dragon health: " + dragon.health );
    }
}
```

Object variables can be **set to null explicitly**, loosing the reference to the object.

```
public class Main {

    public static void main( String[] args )
    {
        Monster troll = new Monster();

        troll = null;

        System.out.println( troll.strength );
    }
}
```

NullPointerException!!