11/20/2019

Thomas Short

CIS 260

Objects and classes

Chapter 9

**Checkpoints**

**9.3.1 Describe the relationship between an object and it’s defining class.**

A defining class is like a blueprint, it gives us an idea of the object we are going to create.

The houses on a street may have the same class or blueprint but the houses may have subtle difference in the way they are built or created.

The objects we create may vary but a circle class should certainly draw a circle, and a class for a square should draw a square because that’s our blueprint or defining class.

**9.3.4 How do you create an object.**

To create an object, you must use a constructor along with the new operator. Typically, we assign the object to a reference variable.

new Circle();

new Square();

**9.4.5 What are the differences between constructors and methods?**

1. A constructor is a special method that is used when we create a new object referred to as an instantiation.

2. Constructors do not have a return type not even void. Constructors have the exact name as the name of the class.

3. Constructors are automatically called when you create a new object, whereas, methods must be called using a method signature

**9.5.1 Which operator is used to access a data field or invoke a method from an object?**

The dot operator(.), also known as the, Object Member Access Operator.

Example = myCircle.radius

myCircle.area

9.5.3 What is NullPointerException?

NullPointerException is a runtime error that occurs when you invoke a method on a reference variable with a null value. It is important to assign an object reference to a variable before invoking a method through a reference variable.

9.5.5 What is wrong with each of the following programs?

**A**. public class ShowErrors {

public static void main(String[] args) {

ShowErrors t = new ShowErrors(5);

}

}

1. **Parameters do no match because the default constructor does not accept integer data types**

**B.** public class ShowErrors {

public static void main(String[] args) {

ShowErrors t = new ShowErrors();

t.x();

}

}

**B. There is no method x() in the show errors class**

1. public class ShowErrors {

public void method1() {

Circle c;

System.out.println("What is radius "

+ c.getRadius());

c = new Circle();

}

}

**C. Runtime error because c is null when it is displayed**

**D**. public class ShowErrors {

public static void main(String[] args) {

C c = new C(5.0);

System.out.println(c.value);

}

}

class C {

int value = 2;

}

1. **they are trying to construt a double object named c but, c’s constructor method**

**only accepts integer values.**

9.6.1 How do you create a Date for the current time? How do you display the current time?

Java has a no-arg constructor in the date class java.util.Date currentDate =new java.util.Date(); then call we date.getTime(); to print convert to a string Date.toString();.

**java.util.Date currentDate = new java.util.Date();**

**currentDate.getTime();**

**System.out.println(currentDate.toString());**

Tue Nov 19 20:54:13 EST 2019

9.7.2 Add the static keyword in the place of? if appropriate

public class Test {

int count; +

public **static** void main(String[] args) {

...

}

public ? int getCount() {

return count;

}

public **static** int factorial(int n) {

int result = 1;

for (int i = 1; i <= n; i++)

result \*= i;

return result;

}

}

9.9.1 What is an accessor method? What is a mutator method? What are the naming conventions for accessor methods and mutator methods?

These methods are derived from the privet modifiers, to access data we want to protect. An accessor method gets data or accesses data we otherwise wouldn’t have access to because it’s privet. A mutator method is a way for us to modify data that is also privet and we would like to protect.

Naming conventions would be as follows:

Getter method

Modifier returnType getPropertyName(); E.g public double getRadius();

{return radius}

If returnType is Boolean:

Public Boolean isPropertyName();

Setter method:

public void setPropertyName(datatype, propertyValue)

E.g. public void setRadius(double, newRadius)

**9.10.3 Show the output of the following code:**

**A.** public class Test {

public static void main(String[] args) {

int[] a = {1, 2};

swap(a[0], a[1]);

System.out.println("a[0] = " + a[0]

+ " a[1] = " + a[1]);

}

public static void swap(int n1, int n2) {

int temp = n1;

n1 = n2;

n2 = temp;

}

}

**A. a[0] = 1 a[1] = 2**

**B.** public class Test {

public static void main(String[] args) {

int[] a = {1, 2};

swap(a);

System.out.println("a[0] = " + a[0]

+ " a[1] = " + a[1]);

}

public static void swap(int[] a) {

int temp = a[0];

a[0] = a[1];

a[1] = temp;

}

}

**B. a[0] = 2 a[1] = 1**

**C**. public class Test {

public static void main(String[] args) {

T t = new T();

swap(t);

System.out.println("e1 = " + t.e1

+ " e2 = " + t.e2);

}

public static void swap(T t) {

int temp = t.e1;

t.e1 = t.e2;

t.e2 = temp;

}

}

class T {

int e1 = 1;

int e2 = 2;

}

**C. e1 = 2 e2 = 1**

**D.** public class Test {

public static void main(String[] args) {

T t1 = new T();

T t2 = new T();

System.out.println("t1's i = " +

t1.i + " and j = " + t1.j);

System.out.println("t2's i = " +

t2.i + " and j = " + t2.j);

}

}

class T {

static int i = 0;

int j = 0;

T() {

i++;

j = 1;

}

}

**D. t1's i = 2 and j = 1**

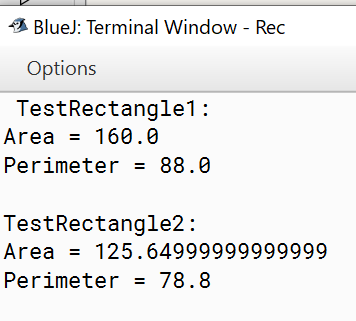
**t2's i = 2 and j = 1**

9.14.1 Describe the roll of **This** keyword.

This reference is needed to access a data field that is normally hidden.

This keyword is the name of a reference that an object can use to refer to itself.

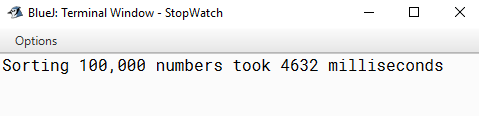
The example of phone using this keyword allows us to reference any instance of the class we are calling or trying to access

Page 362: #9.1 (Rectangle class)

|  |
| --- |
| Rectangle |
| +Rectangle()  +Rectangle(double, double)  +getArea(): double  +getPerimeter(): double |

Page 363: #9.6 (Stopwatch)

|  |
| --- |
| StopWatch |
| +StopWatch()  +start(): void  +stop(): void  +getStartTime(): long  +getEndTime(): long  getElapsedTime(): long |



Page 364: #9.8 (Fan)

|  |
| --- |
| Fan |
| +SLOW: int  +MEDIUM: int  +FAST: int  -fanColor: String  -fanSpeed: int  -fanON: Boolean  -fanradius: double |
| +Fan()  +toString(): String  +isON(): boolean  +setColor(String): void  +getColor(): String  +getRadius(): double  +setRadius(double): void  +setSpeed(int): void  +setOn(boolean): void  +getSpeed(): int |

