*CSE 102*

**Classes (without methods)**

* A class is a blueprint of all objects that will be created from it. An object is a conceptual unit which holds state information together with a set of well-defined operations (methods) that operate on the state.
* Anything in Java is wrapped by a class. You cannot define any function or variable outside a class.
* In Java, each public class has its own file.
* In Java, except primitive data-types (whose first letter is not capitalized) every data-type is associated with a class. By convention class names begin with a capital letter.
* Without any methods, a class is like multiple data-types bundled together to form a new data-type. So it’s like you creating and naming your own data-types. This is something quite useful.
* You can access members of an instance using the dot operator like you have already been doing with .length for instance.

**Exercises**:

1. What will be the output of the following program?

**class** Point {

**double** x;

**double** y;

}

**public** **class** Main

{

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

Point p = **new** Point();

System.***out***.println(p.x);

}

}

1. Compiler Error
2. Runtime Error
3. Some random garbage value
4. 0.0
5. What will be the output of the following program?

**class** Person {

String name;

String surname;

}

**public** **class** Main

{

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

Person p = **new** Person();

p.name = "Niyazi";

System.***out***.println(p.surname);

}

}

1. Compiler Error
2. Niyazi
3. null
4. The function *getYoungest()* is supposed to find out and return the youngest person in an array of people. What should be replaced with the comment?

**class** Person {

String name;

**int** age;

}

**public** **class** Main

{

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

Person p1 = **new** Person();

p1.name = "Fatih";

p1.age = 15;

Person p2 = **new** Person();

p2.name = "Abdullah";

p2.age = 30;

Person p3 = **new** Person();

p3.name = "Ali";

p3.age = 12;

Person[] arr = {p1, p2, p3};

Person youngest = *getYoungest*(arr);

System.***out***.println(youngest.name);

}

**static** Person getYoungest(Person[] people) {

**int** curMin = Integer.***MAX\_VALUE***;

Person res = **null**;

**for**(Person p: people)

**if**(p.age < curMin) {

// here

}

**return** res;

}

}

1. res = p;
2. curMin = p.age;
3. res = p; curMin = p.age;
4. res = p, curMin = p.age;
5. What will be the output of the following program?

**class** Point {

**double** x;

**double** y;

}

**public** **class** Main

{

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

Point p;

Point r = **new** Point();

p = r;

p.x = 3.2;

r.x = 1.1;

System.***out***.println(p.x);

}

}

1. Compiler Error
2. 1.1
3. 3.2
4. Runtime Error
5. The following function *getArea()* is supposed to calculate the area of a given rectangle. What should be replaced with the comment?

**class** Point {

**double** x;

**double** y;

}

**class** Rectangle {

Point topLeft;

Point downRight;

}

**public** **class** Main

{

**static** **double** getArea(Rectangle rect) {

**double** width = rect.downRight.x - rect.topLeft.x;

**double** height = // here

**return** width \* height;

}

}

1. rect.topLeft.y - rect.downRight.y;
2. rect.downRight.y - rect.topLeft.y;
3. rect.topLeft.x - rect.downRight.x;
4. rect.downRight.y + rect.topLeft.y;

Consider the following class definitions. A point is specified with its coordinates and a line is specified by two points that it passes through. Answer the following questions accordingly.

**class** Point {

**double** x;

**double** y;

}

**class** Line {

Point p1;

Point p2;

}

1. Write a function which takes two lines and returns true iff the lines are parallel. You can assume that the lines are not ill-formed (p1 and p2 are distinct points).
2. Complete the *getDist()* function so that it calculates the Euclidean distance between two points given as its parameters.

**static** **double** getDist(Point p1, Point p2) {

}

1. Complete the *isSame()* function so that it returns true iff its parameters represent the same line.

**static** **double** isSame(Line l1, Line l2) {

}

**ANSWERS:**

1. D
2. C
3. C
4. B
5. A

**static** **double** slope(Line l) {

**return** (l.p2.y - l.p1.y) / (l.p2.x - l.p1.x);

}

**static** **boolean** isParallel(Line l1, Line l2) {

**return** *slope*(l1) == *slope*(l2);

}

**static** **double** getDist(Point p1, Point p2) {

**return** Math.*sqrt*(Math.*pow*(p1.x - p2.x, 2) + Math.*pow*(p1.y - p2.y, 2));

}

**static** **double** slope(Line l) {

**return** (l.p2.y - l.p1.y) / (l.p2.x - l.p1.x);

}

**static** **boolean** isParallel(Line l1, Line l2) {

**return** *slope*(l1) == *slope*(l2);

}

**static** **boolean** isEqual(Point p1, Point p2) {

**return** p1.x == p2.x && p1.y == p2.y;

}

**static** **boolean** isSame(Line l1, Line l2) {

**if**(*isEqual*(l1.p1, l2.p1))

**if**(*isEqual*(l1.p2, l2.p2))

**return** **true**;

**else** {

Line l3 = **new** Line();

l3.p1 = l1.p2;

l3.p2 = l2.p2;

**return** *isParallel*(l1, l2) && *isParallel*(l1, l3);

}

**else** {

Line l3 = **new** Line();

l3.p1 = l1.p1;

l3.p2 = l2.p1;

**return** *isParallel*(l1, l2) && *isParallel*(l1, l3);

}

}