Name: SHADAB IQBAL

ID: 19101072 Sec: 1 (set-B)

Ans. to Q No - 1

B

UDP stands for User Datagram Protocol.

When to use TCP: When you need every bit of data and can't afford to lose even 1 packet. ~~Such as~~ For example, in case of download a file or ~~St~~ receiving mail, etc.

When to use UDP: When receiving every bit of data is not a concern. Rather, real-time transmission of data is important. For example, Video Conferencing, media streaming, etc.

Ans. to Q No - 2

A

No, the program won't compile. Because, we are trying to overload the method "doSomething", but methods can not be overloaded just by changing the return type as return type is not a method signature.

~~B~~ C

Design patterns are ~~spefi~~ specific structures of developing a program which can be reused and solves the most frequently occuring problems during software desingning. We use them to speed up the development process as these patterns are well-tested by experienced object oriented software developers.

~~*~~ Name of 3 design patterns are:

1) Factory pattern
2) Singleton pattern
3) Facade pattern.

D

```
String str= sc.nextLine();
String[] words = str.split(" ");
Map<String, Integer> m = new HashMap<>();

Integer c = null;
for (int i=0; i< words.length(); ++i){
    c= m.get(words[i]);
    if (c== null){
        m.put (words[i], 1);
    } else {
        c++;
        m.put(words[i], c);
    }
}
for (Map.Entry<String, Integer> s: m.entrySet()){
    System.out.println(s.getKey() + " "
                            + s.getValue());

}
```

Ans to Q No-3

a

```
public interface Animal {
    public void walk();
    public void eat();
}

abstract class Pet {
    private String name = null;
    public String getName() {
        ~~return name;~~
    }
    public void setName(String name){
~~this.name = name;~~
    }
    public void play() {
    }
}
```

```
public class Spider extends Pet implements Animal {

    public void eat() {

    }
    public void walk() {

    }
}

public class Cat extends Pet implements Animal {

    public Cat (String name){

    }

    public cat () {

    }

    public void play () {

    }

    public void eat() {

    }

    public void walk () {

    }

}
```

```
public class Fish extends Pet implements Animal {

    public Fish (String name) {
    }

    public Fish() {
    }

    public void eat(){
    }

    public void walk() {
    }

}
```

b

In pet class:

```
    private String name= null;

    public String getName() {
        if (name == null) return "Garfield";
        return name;
    }
```

```
public void SetName (String name){
    this.name = name;
}
public void play () {
    system.out.println (getName () + " is playing now.");
}
```

In Cat class:

```
public Cat( String name) {
    setName (name);
}
public void eat() {
    System.out.println("getName()+ " is eating now!");
}
public void walk () {
    System.out.println (getName() + " is walking
    now");
}
```

In Fish Class:

```
public Fish(String name) {
    setName(name);
}

public void walk() {
    System.out.println("Fish cannot walk");
}
```

In Spider class:

```
public void eat() {
    System.out.println("Spider is eating now.");
}
public void walk() {
    System.out.println("Animal with 8 legs
                               is walking.");

}
```