برنامه زمان بندی کارگاه کلادسیم

ساعت	عنوان
۹:۳۰-۹	استقرار و آزمون تعیین سطح
۱۰:۳۰-۹:۳۰	رایانش ابری بصورت فشرده
۱۰:۴۵-۱۰:۳۰	استراحت
۱۲:۳۰-۱۰:۴۵	استقرار و کار عملی با کلادسیم (بخش اول)
۱۲:۳۰ تا ۱۴	استراحت (نماز و نهار)
۱۴:۳۰-۱۶	کار عملی با کلادسیم (بخش دوم)

مراحل استفاده از کلادسیم:

۱- فایل Eclipse.zip را از حالت فشرده خارج کنید.

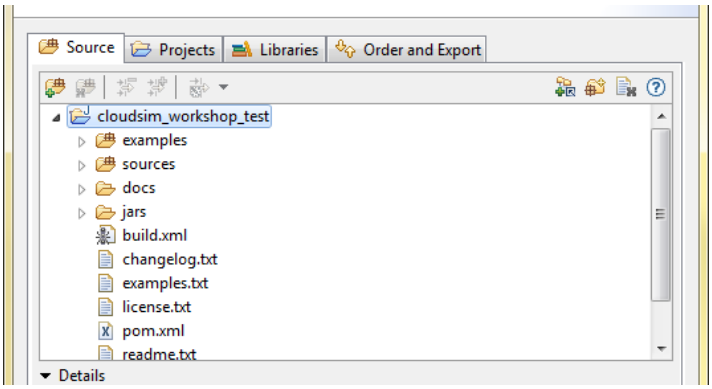
۲- فایلcloudsim3.zipرا از حالت فشرده خارج کنید

۳- فایلcommons-math3-3.2.zip را از حالت فشرده خارج کنید

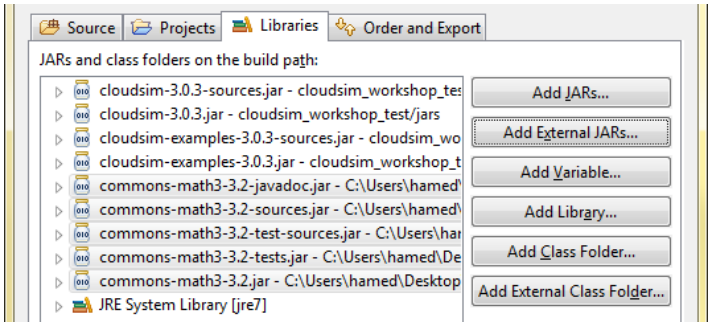
۴- برنامه Eclipse را اجرا کنید.

۵- File>New > Project> Java Project را انتخاب نمایید > Next

۶- نام پروژه خود را وارد کنید و مسیر cloudsim-3.0.3 را در Location وارد نمایید > Next



۷- در بخش Libraries وارد شوید > Add External JARs > کلیه فایل های jar پوشه common-math3-3.2 را اضافه نمایید > Finish



۸- فایل examples\org.cloudbus.cloudsim.examples\CloudSimExample1.java را باز کنید > Run

در صورتیکه نیاز به تنظیم SDK بود بصورت زیر عمل کنید:

a. به Start> Computer> Properties>Advanced system Settings>Advanced>Environment Variable مراجعه کنید

b. مسیر JDK را در متغییر JAVA_HOME در بخش User Variables اضافه کنید.

```
// A simple example showing how to create a datacenter with one host and run one cloudlet on it.

public class CloudSimExample1 {

	private static List<Cloudlet> cloudletList;
	private static List<Vm> vmlist;

	public static void main(String[] args) {

		// First step: Initialize the CloudSim package. It should be called before creating any entities.
		int num_user = 1;                                   // number of cloud users
		Calendar calendar = Calendar.getInstance();
		boolean trace_flag = false;                         // mean trace events

		CloudSim.init(num_user, calendar, trace_flag);      // Initialize the CloudSim library

		// Second step: Create Datacenters
		// Datacenters are the resource providers in CloudSim. We need at list one of them to run a CloudSim simulation
		Datacenter datacenter0 = createDatacenter("Datacenter_0");

		DatacenterBroker broker = createBroker();           // Third step: Create Broker
		int brokerId = broker.getId();

		vmlist = new ArrayList<Vm>();                       // Fourth step: Create one virtual machine

		// VM description
		int vmid = 0;
		int mips = 1000;
		long size = 10000;                                  // image size (MB)
		int ram = 512;                                      // vm memory (MB)
		long bw = 1000;
		int pesNumber = 1;                                  // number of cpus
		String vmm = "Xen";                                 // VMM name

		Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared()); // create VM

		vmlist.add(vm);                                     // add the VM to the vmList

		broker.submitVmList(vmlist);                        // submit vm list to the broker

		cloudletList = new ArrayList<Cloudlet>();           // Fifth step: Create one Cloudlet

		// Cloudlet properties
		int id = 0;
		long length = 400000;
		long fileSize = 300;
		long outputSize = 300;
		UtilizationModel utilizationModel = new UtilizationModelFull();

		Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
		cloudlet.setUserId(brokerId);
		cloudlet.setVmId(vmid);

		cloudletList.add(cloudlet);                         // add the cloudlet to the list

		broker.submitCloudletList(cloudletList);            // submit cloudlet list to the broker

		CloudSim.startSimulation();                         // Sixth step: Starts the simulation
		CloudSim.stopSimulation();

		List<Cloudlet> newList = broker.getCloudletReceivedList();  //Final step: Print results when simulation is over
		printCloudletList(newList);

		datacenter0.printDebts();                           // Print the debt of each user to each datacenter
	}
```

```
    // Creates the datacenter.
    private static Datacenter createDatacenter(String name) {
        List<Host> hostList = new ArrayList<Host>();                    // 1. We need to create a list to store our machine

        // 2. A Machine contains one or more PEs or CPUs/Cores. In this example, it will have only one core.
        List<Pe> peList = new ArrayList<Pe>();

        int mips = 1000;

        // 3. Create PEs and add these into a list.
        peList.add(new Pe(0, new PeProvisionerSimple(mips)));           // need to store Pe id and MIPS Rating

        // 4. Create Host with its id and list of PEs and add them to the list of machines
        int hostId = 0;
        int ram = 2048;                                                 // host memory (MB)
        long storage = 1000000;                                         // host storage
        int bw = 10000;

        hostList.add(
            new Host(hostId,new RamProvisionerSimple(ram),new BwProvisionerSimple(bw),storage,peList
                ,new VmSchedulerTimeShared(peList))
            );                                                          // This is our machine

        // 5. Create a DatacenterCharacteristics object that stores the properties of a data center: architecture, OS, list of
        // Machines, allocation policy: time- or space-shared, time zone and its price (G$/Pe time unit).
        String arch = "x86";                                            // system architecture
        String os = "Linux";                                            // operating system
        String vmm = "Xen";
        double time_zone = 10.0;                                        // time zone this resource located
        double cost = 3.0;                                              // the cost of using processing in this resource
        double costPerMem = 0.05;                                       // the cost of using memory in this resource
        double costPerStorage = 0.001;                                  // the cost of using storage in this resource
        double costPerBw = 0.0;                                         // the cost of using bw in this resource
        LinkedList<Storage> storageList = new LinkedList<Storage>();    // we are not adding SAN devices by now

        DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                arch, os, vmm, hostList, time_zone, cost, costPerMem,  costPerStorage, costPerBw);

        Datacenter datacenter = null;                                   // 6. Finally, we need to create a PowerDatacenter object.
        datacenter = new Datacenter(name, characteristics, new VmAllocationPolicySimple(hostList), storageList, 0);

        return datacenter;
    }
    // Creates the broker.
    private static DatacenterBroker createBroker() {
        DatacenterBroker broker = null;
        broker = new DatacenterBroker("Broker");
        return broker;
    }
    // Prints the Cloudlet objects.
    private static void printCloudletList(List<Cloudlet> list) {
        int size = list.size();
        Cloudlet cloudlet;
        String indent = "    ";
        Log.printLine();
        Log.printLine("========== OUTPUT ==========");
        Log.printLine("Cloudlet ID" + indent + "STATUS" + indent
                + "Data center ID" + indent + "VM ID" + indent + "Time" + indent+ "Start Time" + indent + "Finish Time");
        DecimalFormat dft = new DecimalFormat("###.##");
        for (int i = 0; i < size; i++) {
            cloudlet = list.get(i);
            Log.print(indent + cloudlet.getCloudletId() + indent + indent);
            if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
                Log.print("SUCCESS");
                Log.printLine(indent + indent + cloudlet.getResourceId()
                                        + indent + indent + indent + cloudlet.getVmId()
                                        + indent + indent
                                        + dft.format(cloudlet.getActualCPUTime()) + indent
                                        + indent + dft.format(cloudlet.getExecStartTime())
                                        + indent + indent
                                        + dft.format(cloudlet.getFinishTime()));
            }
        }
    }
}
```

پروفایل تمرین ۱

تعداد مرکز داده : ۱

معماری: x86

سیستم عامل: Linux

فوق ناظر: Xen

محدوده زمانی: 3.5+

هزینه پردازنده: ۱۰ تومان

هزینه حافظه: ۲ تومان

هزینه دیسک: ۲ تومان

هزینه پهنای باند: ۱۰ تومان

تعداد ماشین میزبان: ۱

توان پردازشی: 1000 mips

تعداد هسته پردازشی: ۱

میزان رم: ۲۰۴۸ مگابایت

دیسک: ۱۰۰۰۰۰۰ مگابایت

پهنای باند: ۱۰۰۰۰ مگابیت

تعداد کاربر: ۱

تعداد درخواست ماشین مجازی: ۱

توان پردازشی: 1000 mips

اندازه ماشین: 10000 مگابایت

اندازه رم: 512 مگابایت

پهنای باند: 1000 مگابیت

تعداد پردازنده: ۱

فوق ناظر: Xen

تعداد کار بر روی ماشین مجازی: ۱

طول درخواست: 400000 دستورالعمل

اندازه فایل ورودی: 300 مگابایت

اندازه فایل خروجی: 300 مگابایت

مدل بارکاری: ۱۰۰ درصد

پروفایل تمرین ۲

تعداد مرکز داده : ۱

معماری: x86

سیستم عامل: Linux

فوق ناظر: Xen

محدوده زمانی: 3.5+

هزینه پردازنده: ۱۰ تومان

هزینه حافظه: ۲ تومان

هزینه دیسک: ۲ تومان

هزینه پهنای باند: ۱۰ تومان

تعداد ماشین میزبان: ۱

توان پردازشی: 1000 mips

تعداد هسته پردازشی: ۱

میزان رم: ۲۰۴۸ مگابایت

دیسک: ۱۰۰۰۰۰۰ مگابایت

پهنای باند: ۱۰۰۰۰ مگابیت

تعداد کاربر: ۱

تعداد درخواست ماشین مجازی: ۱

توان پردازشی: 1000 mips

اندازه ماشین: 10000 مگابایت

اندازه رم: 512 مگابایت

پهنای باند: 1000 مگابیت

تعداد پردازنده: ۱

فوق ناظر: Xen

تعداد کار بر روی ماشین مجازی: ۲

طول درخواست: 400000 دستورالعمل اندازه فایل ورودی: 300 مگابایت اندازه فایل خروجی: 300 مگابایت مدل بارکاری: ۱۰۰ درصد	طول درخواست: 400000 دستورالعمل اندازه فایل ورودی: 300 مگابایت اندازه فایل خروجی: 300 مگابایت مدل بارکاری: ۱۰۰ درصد

سئوال۲-۱: چه تغییراتی برای افزودن کار دوم مورد نیاز است؟

پروفایل تمرین ۳

تعداد مرکز داده : ۱

معماری: x86

سیستم عامل: Linux

فوق ناظر: Xen

محدوده زمانی: 3.5+

هزینه پردازنده: ۱۰ تومان

هزینه حافظه: ۲ تومان

هزینه دیسک: ۲ تومان

هزینه پهنای باند: ۱۰ تومان

تعداد ماشین میزبان: ۱

توان پردازشی: 1000 mips

تعداد هسته پردازشی: ۱

میزان رم: ۲۰۴۸ مگابایت

دیسک: ۱۰۰۰۰۰۰ مگابایت

پهنای باند: ۱۰۰۰۰ مگابیت

تعداد کاربر: ۱

تعداد درخواست ماشین مجازی: ۲

توان پردازشی: 500 mips اندازه ماشین: 10000 مگابایت اندازه رم: 512 مگابایت پهنای باند: 1000 مگابیت تعداد پردازنده: ۱ فوق ناظر: Xen	توان پردازشی: 500 mips اندازه ماشین: 10000 مگابایت اندازه رم: 512 مگابایت پهنای باند: 1000 مگابیت تعداد پردازنده: ۱ فوق ناظر: Xen

تعداد کار بر روی ماشین مجازی: ۲

طول درخواست: 400000 دستورالعمل اندازه فایل ورودی: 300 مگابایت اندازه فایل خروجی: 300 مگابایت مدل بارکاری: ۱۰۰ درصد بر روی ماشین مجازی اول اجرا شود.	طول درخواست: 400000 دستورالعمل اندازه فایل ورودی: 300 مگابایت اندازه فایل خروجی: 300 مگابایت مدل بارکاری: ۱۰۰ درصد بر روی ماشین مجازی دوم اجرا شود.

سئوال۳-۱: چه تغییراتی برای افزودن ماشین مجازی دوم مورد نیاز است؟

سئوال۳-۲: در صورت استفاده از توان پردازشی ۵۰۰ یا ۱۰۰۰ در ماشین های مجازی، چه تفاوتی در روند اجرا رخ میدهد؟

پروفایل تمرین ۴

تعداد مرکز داده : ۱

معماری: x86

سیستم عامل: Linux

فوق ناظر: Xen

محدوده زمانی: 3.5+

هزینه پردازنده: ۱۰ تومان

هزینه حافظه: ۲ تومان

هزینه دیسک: ۲ تومان

هزینه پهنای باند: ۱۰ تومان

تعداد ماشین میزبان: ۱

توان پردازشی: 1000 mips

تعداد هسته پردازشی: ۱

میزان رم: ۲۰۴۸ مگابایت

دیسک: ۱۰۰۰۰۰۰ مگابایت

پهنای باند: ۱۰۰۰۰ مگابیت

تعداد کاربر: ۲

تعداد درخواست ماشین مجازی: ۲

توان پردازشی: 500 mips اندازه ماشین: 10000 مگابایت اندازه رم: 512 مگابایت پهنای باند: 1000 مگابیت تعداد پردازنده: ۱ فوق ناظر: Xen **متعلق به کاربر اول**	توان پردازشی: 500 mips اندازه ماشین: 10000 مگابایت اندازه رم: 512 مگابایت پهنای باند: 1000 مگابیت تعداد پردازنده: ۱ فوق ناظر: Xen **متعلق به کاربر دوم**

تعداد کار بر روی ماشین مجازی: ۲

طول درخواست: 400000 دستورالعمل اندازه فایل ورودی: 300 مگابایت اندازه فایل خروجی: 300 مگابایت مدل بارکاری: ۱۰۰ درصد **متعلق به کاربر اول**	طول درخواست: 400000 دستورالعمل اندازه فایل ورودی: 300 مگابایت اندازه فایل خروجی: 300 مگابایت مدل بارکاری: ۱۰۰ درصد **متعلق به کاربر دوم**

سئوال۴-۱: چه تغییراتی برای افزودن کاربر دوم مورد نیاز است؟

پروفایل تمرین ۵

تعداد مرکز داده : ۱

معماری: x86

سیستم عامل: Linux

فوق ناظر: Xen

محدوده زمانی: 3.5+

هزینه پردازنده: ۱۰ تومان

هزینه حافظه: ۲ تومان

هزینه دیسک: ۲ تومان

هزینه پهنای باند: ۱۰ تومان

تعداد ماشین میزبان: ۲

توان پردازشی: 1000 mips	توان پردازشی: 1000 mips
تعداد هسته پردازشی: ۱	تعداد هسته پردازشی: ۱
میزان رم: ۲۰۴۸ مگابایت	میزان رم: ۲۰۴۸ مگابایت
دیسک: ۱۰۰۰۰۰۰ مگابایت	دیسک: ۱۰۰۰۰۰۰ مگابایت
پهنای باند: ۱۰۰۰۰ مگابیت	پهنای باند: ۱۰۰۰۰ مگابیت

تعداد کاربر: ۲

تعداد درخواست ماشین مجازی: ۲

توان پردازشی: 1000 mips	توان پردازشی: 1000 mips
اندازه ماشین: 10000 مگابایت	اندازه ماشین: 10000 مگابایت
اندازه رم: 512 مگابایت	اندازه رم: 512 مگابایت
پهنای باند: 1000 مگابیت	پهنای باند: 1000 مگابیت
تعداد پردازنده: ۱	تعداد پردازنده: ۱
فوق ناظر: Xen	فوق ناظر: Xen
متعلق به کاربر اول	**متعلق به کاربر دوم**

تعداد کار بر روی ماشین مجازی: ۲

طول درخواست: 400000 دستورالعمل	طول درخواست: 400000 دستورالعمل
اندازه فایل ورودی: 300 مگابایت	اندازه فایل ورودی: 300 مگابایت
اندازه فایل خروجی: 300 مگابایت	اندازه فایل خروجی: 300 مگابایت
مدل بارکاری: ۱۰۰ درصد	مدل بارکاری: ۱۰۰ درصد
متعلق به کاربر اول	**متعلق به کاربر دوم**

سئوال۵-۱: چه تغییراتی برای افزایش تعداد منابع در مرکز داده مورد نیاز است؟

سئوال۵-۲: چه تفاوتی در روند اجرای ماشین های مجازی با توان پردازشی ۱۰۰۰ یا ۵۰۰ وجود خواهد داشت؟

پروفایل تمرین ۶

تعداد مرکز داده : ۲

معماری: x86	معماری: x86
سیستم عامل: Linux	سیستم عامل: Linux
فوق ناظر: Xen	فوق ناظر: Xen
محدوده زمانی: 3.5+	محدوده زمانی: 3.5+
هزینه پردازنده: ۱۰ تومان	هزینه پردازنده: ۱۰ تومان
هزینه حافظه: ۲ تومان	هزینه حافظه: ۲ تومان
هزینه دیسک: ۲ تومان	هزینه دیسک: ۲ تومان
هزینه پهنای باند: ۱۰ تومان	هزینه پهنای باند: ۱۰ تومان

تعداد ماشین میزبان در هر مرکز داده: ۱

توان پردازشی: 1000 mips
تعداد هسته پردازشی: ۱
میزان رم: ۲۰۴۸ مگابایت
دیسک: ۱۰۰۰۰۰۰ مگابایت
پهنای باند: ۱۰۰۰۰ مگابیت

تعداد کاربر: ۱

تعداد درخواست ماشین مجازی: ۲

توان پردازشی: 1000 mips	توان پردازشی: 1000 mips
اندازه ماشین: 10000 مگابایت	اندازه ماشین: 10000 مگابایت
اندازه رم: 512 مگابایت	اندازه رم: 512 مگابایت
پهنای باند: 1000 مگابیت	پهنای باند: 1000 مگابیت
تعداد پردازنده: ۱	تعداد پردازنده: ۱
فوق ناظر: Xen	فوق ناظر: Xen

تعداد کار بر روی ماشین مجازی: ۲

طول درخواست: 400000 دستورالعمل	طول درخواست: 400000 دستورالعمل
اندازه فایل ورودی: 300 مگابایت	اندازه فایل ورودی: 300 مگابایت
اندازه فایل خروجی: 300 مگابایت	اندازه فایل خروجی: 300 مگابایت
مدل بارکاری: ۱۰۰ درصد	مدل بارکاری: ۱۰۰ درصد

سئوال۶-۱: چه تغییراتی برای افزودن مرکز داده دوم مورد نیاز است؟

سئوال۶-۲: تخصیص کار کاربر به هر مرکز داده در چه بخشی از کد انجام می شود؟