# 1. Програм болон арга хоёрын ялгааг өөрийн үгээр тайлбарлана уу.

Арга гэдэг нь ихэвчлэн тодорхой үйлдлийг гүйцэтгэдэг хэллэгүүдийн нэрлэгдсэн цуглуулга юм. Харин програм нь хэрэглэгчийн өмнөөс даалгавар гүйцэтгэдэг. Жишээ нь ACM Java санг ашиглах үед програм нь acm.program багцад тодорхойлогдсон байна.

# 2. Аргын хувьд хэрэглэдэг call-дуудах, argument-аргумент, return-буцаах гэсэн нэр томъёог тайлбарлана уу.

Тодорхой аргаар гүйцэтгэсэн үйлдлийг дуудах шаардлагатай үед нэрийг нь зааж, дараа нь хаалтанд оруулсан утгуудын жагсаалтыг дуудна. Java нь аргын дуудлагыг гүйцэтгэх үед дуудлагын аргыг түр зогсоож, дуудагдсан аргын эхний мэдэгдлээс эхлэн гүйцэтгэлийг үргэлжлүүлнэ.

Хаалтанд байгаа утгыг аргумент гэж нэрлэдэг; эдгээр утгыг аргын зарласан харгалзах албан ёсны параметрийн хувьсагчид хуулж авна.

Аргын гүйцэтгэл дуусмагц дуудаж буй програм руу буцах бөгөөд энэ нь дууссан арга болон түүний локал хувьсагчууд алга болж, үүний дараа Java өмнө нь түр зогсоосон аргыг үргэлжлүүлэн ажиллуулна гэсэн үг юм.

# 3. Аргументийг ашиглан арга руу мэдээлэл дамжуулах болон readInt гэх мэтийн аргыг аргыг ашиглан оролтын өгөгдлийг унших хоёрын ялгаа нь юу вэ? Хэзээ  үйлдэл бүр зохистой вэ?

Аргументууд нь нэг аргыг нөгөөд нь мэдээлэл дамжуулах механизмыг бий болгодог. Харин readInt гэх мэт оролтын аргууд нь хэрэглэгч програмтай мэдээлэл дамжуулах механизмыг хангадаг. Жишээлбэл, хэрэглэгчийн оруулсан бүхэл тоонуудын жагсаалтыг нэмэх программ зохиогдсон бол тэдгээр утгыг readInt ашиглан унших болно. Гэсэн хэдий ч, хэрэв та илүү төвөгтэй тооцооллын нэг хэсэг болгон квадрат язгуурыг тооцоолох шаардлагатай бол оронд нь тохирох утгыг аргумент болгон Math.sqrt аргад шилжүүлэх болно.

# 4. Java-д аргын үр дүн гэдгийг хэрхэн тодорхойлж өгдөг вэ?

Аргын үр дүнг үндсэн хэсэгт буцаахын тулд ***return statement*** ашигладаг. Хүчингүй гэж зарласан аливаа арга нь утгыг буцаадаггүй. Энэ нь буцаах мэдэгдлийг агуулсан байх шаардлагагүй.

# 5. Аргын бие хэсэг дотор нэгээс олон return заавар байж болох уу?

Аргын бие хэсэгт хэдэн ч буцах мэдэгдэл байж болно

# 6. Өөр нэгэн объектод аргыг хэрэглэнэ гэдгийг илтгэн харуулахын тулд юу хийх вэ?

Өөр нэгэн объектод аргыг хэрэглэнэ гэдгийг илтгэн харуулахын тулд

***receiver.method(arguments)***

# 7. Өмнөх хичээлд үзүүлсэн ихэнх програмд яагаад хүлээн авагчийг заавал тодорхойлох/нарийвчлан зааж өгөх шаардлаггүй байсан вэ?

Хэрэв аргын хувьд хүлээн авагчийг заагаагүй бол Java тухайн аргыг одоогийн объектод хэрэглэдэг. Өмнөх бүлгүүдийн програмуудад ихэнх аргуудыг одоогийн объектод ашиглаж байсан.

# 8. Сурах бичгийн 5-р бүлэгт үзүүлсэн monthName гэсэн аргын case өгүүлбэр бүрийн төгсгөлд break заавар яагаад хэрэггүй байсан вэ?

monthName аргын тохиолдол заалтууд нь буцах мэдэгдлээр төгсдөг тул завсарлага шаарддаггүй. Буцах мэдэгдэл нь тухайн үед програмыг гарахад хүргэдэг бөгөөд энэ нь програм дараагийн тохиолдол заалт руу үргэлжлэхгүй гэсэн үг юм.

# 9. Хоёр тэмдэгт мөр ижилхэн тэмдэгт агуулсан байна гэдгийг хэрхэн олж тогтоох вэ?

***equals*** аргыг хэрэглэж, нэг мөрийг аргумент, нөгөөг нь болгон хүлээн авагч ашиглана.

# 10. Аргумент болон формал параметр хоёрын хооронд ямар харилцан уяалдаа байна вэ?

Энэ текст нь аргументууд нь аргын дуудлагын хэсэг болох илэрхийлэлд хамаарах нэр томъёог ашигладаг; албан ёсны параметр гэсэн нэр томъёо нь аргументын утгуудыг хуулсан аргын толгой хэсэгт байгаа хувьсагчийн нэрсийг хэлнэ.

# 11. Аргын дотор зарласан хувьсагчийг дотоод хувьсагч гэж нэрлэдэг. Энд дотоод гэдэг үг ямар утга учиртай вэ?

Дотоод хувьсагчид зөвхөн зарласан аргын дотор л харагдана. Энэ аргын гадна эдгээр хувьсагч нь тодорхойгүй байгаа бөгөөд өөр аргаас тухайн хувьсагчийг ашиглаж болохгүй учир дотоод гэсэн.

# 12. Return address-буцах хаяг гэдэг нэр томъёо ямар утга учиртай вэ?

Буцах хаяг гэдэг нэр томъёо нь дуудлагын аргын цэгийг илэрхийлдэг бөгөөд энэ нь арга буцаж ирэхэд гүйцэтгэл үргэлжлэх болно. Java-н ажиллах цагийн систем нь аргын дуудлага хийх бүртээ буцах хаягийг хадгалах ёстой.

# 13. Алхам алхмаар сайжруулах-stepwise refinement процессийг өөрийн үгээр тайлбарлана уу.

Алхам алхмаар боловсронгуй болгох стратеги нь програмыг энгийн дэд асуудлуудыг дараалан шийдвэрлэх аргууд болгон хуваахаас бүрддэг. Техникийг хэрэгжүүлэхийн тулд та асуудлын хамгийн ерөнхий мэдэгдлээс эхэлдэг. Та эдгээр аргуудын тодорхойлолтыг бичиж амжаагүй байгаа хэдий ч аливаа нарийн төвөгтэй үйлдлүүдийг аргын дуудлагууд болгон төлөөлөх арга хэлбэрээр асуудлын шийдлийг кодлодог. Нэг түвшний задралыг дуусгасны дараа та дэд асуудлуудыг шийдвэрлэх аргуудыг үргэлжлүүлж, ижил процессыг ашиглаж болох бөгөөд энэ нь эргээд шинэ дэд асуудлууд үүсгэж болно. Бүх үйлдлүүд нь нэмэлт хуваагдалгүйгээр кодлоход хангалттай хялбар болтол процесс үргэлжилнэ.

# 14. brute-force - Муйхар-хүчээр гэдэг алгоритмийг юу гэж ойлгох вэ?

***Brute-Force*** алгоритм нь бүх боломжийг туршиж үздэг алгоритм юм.

# 15. 7735, 4185 гэсэн хоёр тооны хамгийн их ерөнхий хуваагчийг Евклидын алгоритмыг хэрэглэн тооцоолсон. Бодолт хийх үед дотоод хувьсагч r ямар утгыг авах вэ?

7735 ба 4185 тоонуудын хамгийн том нийтлэг хуваагч нь 5. Евклидийн алгоритм нь while давталтын дараалсан мөчлөгт r-ийн дараах утгуудыг тооцоолох замаар энэ үр дүнг тодорхойлдог: 3550, 635, 375, 260, 115, 30, 25, 5.

# 16. Евклидын алгоритмын бодох жишээ gcd(x, y)-ийн хувьд x нь y –ээс үргэлж их байсан. Хэрэв x нь y-ээс бага бол яах бол?

Евклидийн алгоритмын код нь x нь у-аас бага байсан ч ажилладаг. Ийм зүйл тохиолдоход while давталтын эхний мөчлөг нь x ба y-ийн утгуудыг сольж дуусна

# 17. Програмчлахад нэг том програм бичвэл ямар давуу болон сул талтай вэ? Яагаад заавал жижиглэн задлах хэрэгтэй вэ? Жижиглэн задалсаны давуу болон сул талыг тайлбарлана уу.

Томоохон программуудыг бүхэлд нь ойлгоход маш хэцүү байдаг. Ийм программуудыг хүний ​​ойлголтын хүрээнд байлгах цорын ганц арга бол тэдгээрийг илүү энгийн, удирдах боломжтой хэсгүүдэд хуваах явдал юм. Задаргааг зөв тодорхойлж чадвал тус бүр нь нэгдмэл ойлголттой байх бөгөөд програмыг бүхэлд нь ойлгоход илүү хялбар болгоно. Хэрэв дэд даалгавруудыг зохисгүй сонговол таны задралд саад болж болзошгүй.

# 18. Java програмчлалын хэлийг ашиглан нэг том системийг ямар ямар хэсгүүдэд хуваах боломжтой вэ?

Аргыг ашиглан тухайн сисиемээ дэд хэсгүүдэд хуваана. Шаардлагатай тохиолдолд тухайн дэд хэсгүүдээ дотор нь дахин дэд сэдвүүдэд хуваах боломжтой.

# 19. Ганц аргын тусламжтайгаар хэрэгжүүлж болохуйц хялбар даалгавар болтол програмыг хуваан жижиглэх энэ процесс буюу арга техникийг юу гэж нэрлэдэг вэ?

Жижиглэн хуваах стратеги буюу decomposition гэнэ.

# 21. Програмыг жижиглэн задалсаны дараа тэдгээр жижиг хэсгүүд хооронд мэдээлэл солилцох эсвэл өөр нэг програмын хэсэгт оршин буй мэдээллийг хэрэглэх шаардлагатай болно. Програмын аль нэгэн хэсэгт өөр нэгэн хэсэгт боловсруулсан эсвэл боловсруулж буй өгөгдлийг ашиглах бол үүнийг хэрхэн хэрэгжүүлэх вэ? Арга замыг тодорхой жишээгээр тайлбарлаж үзүүлнэ үү.

import java.io.\*;

public class exec {

public static void main(String argv[]) {

int i = 5, j = 6, k = 7;

BufferedReader rd = new BufferedReader(new InputStreamReader(System.in));

try {

InputStreamReader isr = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(isr);

System.out.println("Enter class name");

String s = br.readLine();

Process pro = Runtime.getRuntime().exec(s);

BufferedReader in = new BufferedReader(new InputStreamReader(pro.getInputStream()));

String line=null;

while((line = in.readLine()) != null) {

System.out.println(line);

}

in.close();

} catch(Exception err) {

err.printStackTrace();

}

}

}

Та Java програмдаа дараах байдлаар солилцож болно.

String[] cmd = { s, Integer.toString(n) };

Process pro=Runtime.getRuntime().exec(cmd);

# 22. Жижиглэн задалсан хэсгүүдийн хувьд, ямар нөхцөлд тогтмолыг, ямар нөхцөлд аргументыг мэдээлэл хадгалахад хэрэглэвэл зохистой вэ?

Тухайн жижиг хэсэг бүр хоорондоо нийтлэг мэдээлэлтэй байвал нэрлэсэн тогтмолоыг ашиглана. Харин мэдээллүүд нь тус бүртээ ялгаатай байвал аргументийг ашиглана.

Proggramming

5.2

Graphical user interface, application, Word

Description automatically generated

5.5

Icon

Description automatically generated Graphical user interface, text, application

Description automatically generated

5.6

Graphical user interface, text, application

Description automatically generated

5.7

Graphical user interface, text, application

Description automatically generated

5.8

Graphical user interface, text, application

Description automatically generated

5.12

Graphical user interface, text, application

Description automatically generated

Lecture examples

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Lecture 7

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated