জাভা প্রোগ্রামিং

অটোমেটিক স্ক্রলের মাধ্যমে ই-বুক পড়া / রিড়ের জন্যঃ

আপনার ই-বুক বা pdf রিডারের Menu Bar এর View অপশনটি তে ক্লিক করে Auto /Automatically Scroll অপশনটি সিলেক্ট করুন (অথবা সরাসরি যেতে ⇨ Ctrl + Shift + H)। এবার ↑ up Arrow বা ↓ down Arrow তে ক্লিক করে আপনার পড়ার সুবিধা অনুসারে স্ক্রল স্পীড ঠিক করে নিন।

যেহেতু আপনারা কম্পিউটার প্রোগ্রামিং শিখতে যাচ্ছেন, শুরু থেকেই আমি কিছু ব্যাপারে অনুমান করে নিচ্ছি: এই মুহূর্তে আপনি কম্পিউটার প্রোগ্রামিং সম্বন্ধে কিছুই জানেন না। যদি ইতিমধ্যেই আপনি কিছু জেনে থাকেন, এই অনুচ্ছেদের ১ম অংশ আপনার কাছে খুব সোজা মনে হবে। কোন প্রকার সংশয় না করে আপনি যতক্ষন নতুন জিনিস না পান বাদ দিয়ে যেতে পারেন।

আমি এটা অনুমান করে নিচ্ছি যে, আপনি আপনার ব্যবহৃত কম্পিউটার সম্বন্ধে কিছু জানেন। অর্থাৎ আমি অনুমান করে নিচ্ছি আপনি এর মধ্যেই জানেন কিভাবে একটি ফাইল সম্পাদনা করা, অনুকৃতি ও মুছে ফেলা, নতুন নামকরন করা, আপনার কর্মপরিধিতে তথ্য অনুসন্ধান করা ইত্যাদি করতে হয়।

সরলতার জন্য, আমি ধরে নিচ্ছি উইন্ডোজ ৯৫, ৯৮. ২০০, এন.টি. বা এক্সপি চলছে এমন একটি কম্পিউটার ব্যবহার করছেন। আশা করি অন্যান্য অপারেটিং সিস্টেম ব্যবহারকারীদের কাছে ধারণাগুলো মিলানো তুলনামূলকভাবে সহজবোধ্য হবে।

এবং আমি ধরে নিচ্ছি যে শিখার ব্যাপারে আপনার আগ্রহ আছে।
জাভা তে প্রোগ্রামিং শুরু করার জন্য দরকারি জিনিসপত্র ওয়েবে বিনা পয়সায় পাওয়া যায়। তাছাড়া জাভার জন্য প্রচুর পরিমান শিক্ষনীয় বিষয়বস্তু রয়েছে ওয়েবে, সুতরাং এই অনুচ্ছেদ শেষ করা মাত্রই খুব সহজেই আপনি আপনার দক্ষতাকে উন্নত করতে আরও শিখতে পারেন। কম্পাইলার, তৈরী করার জিনিসপত্র, পড়ার বিষয়বস্তু ইত্যাদি ব্যাপারে পয়সা খরচ না করে এখানে আপনি জাভা প্রোগ্রামিং শিখতে পারেন। জাভা শিখার পর অন্যান্য ভাষা(প্রোগ্রামিং) শিখা সহজ হয়ে যায়। তাই শুরু করার জন্য জাভা ভালো একটা বিষয়।

কিছু কথা:

আমি ধরে নিচ্ছি আপনি প্রোগ্রামিং সম্বন্ধে কিছুই জানেন না - এটা মাথায় রাখবেন। সম্পূর্ণ ব্যাপারটা বোঝার জন্য কিছু বিশেষ শব্দ সম্পর্কে জানা দরকার:

কম্পিউটার প্রোগ্রাম - কম্পিউটার প্রোগ্রাম হচ্ছে কিছু নির্দেশনার সমষ্টি যা অনুযায়ী কম্পিউটার পুঙ্খানুপুঙ্খভাবে কাজ করে। কম্পিউটারের প্রতি নির্দেশনাগুলো হতে পারে - কিছু সংখ্যার যোগ, দুটি সংখ্যার তুলনা করা এবং এর ফলাফলের উপর ভিত্তি করে কোন সিদ্ধান্তে উপনিত হওয়া এবং আরও কত কি। কিন্তু কম্পিউটারের কাছে কম্পিউটার প্রোগ্রাম হচ্ছে কিছু নির্দেশনাবলী, যেমন বাবুর্চির কাছে রান্নার প্রস্তুতপ্রণালী বা যেমন সুরকারের কাছে নির্দেশনাবলী হচ্ছে স্বরলিপি। দেনা-পাওনার হিসাব যেভাবে মিলানো হয় বা পর্দায় যেমন খেলা দেখানো হয় বা 'ওয়ার্ড প্রসেসর' "Word Processor" যেভাবে বাস্তবে রূপ দেয়া হয় ঠিক তেমনি কম্পিউটার আপনার নির্দেশনাবলী সঠিকভাবে অনুসরণ করবে এবং এরই মাধ্যমে গুরুত্বপূর্ণ কিছু কাজ করবে।

প্রোগ্রামিং ভাষা - আপনার দেয়া নির্দেশনাবলী ঠিকমতো বোঝার জন্য ঐ নির্দেশনাগুলো এমন এক ভাষায় লিখতে হয় যা কম্পিউটার বুঝতে পারে এবং তা হচ্ছে প্রোগ্রামিং ল্যাংগুয়েজ বা প্রোগ্রামিং ভাষা। মুখে যেমন অনেক ধরনের ভাষা ব্যবহৃত হয় ঠিক তেমনি প্রোগ্রামিং ভাষাও রয়েছে অনেক ধরনের - ফোরট্রান(Fortran), কোবল Cobol), বেসিক(Basic), প্যাসকাল(Pascal), সি(C), সি++(C++), জাভা(Java), পার্ল(Perl)। এই ভাষাগুলো সবাই বিভিন্ন ভাবে মোটামুটি একই ধরনের মতাদর্শ প্রকাশ করে।

কম্পাইলার বা সংকলক - কম্পাইলার মানুষের বোধগম্য কম্পিউটার ভাষায় (যেমন জাভা) লেখা একটি কম্পিউটার প্রোগ্রামকে এমন এক রূপে রূপান্তর করে যা কম্পিউটার চালাতে পারে। হয়ত আপনার কম্পিউটারে আপনি "ইএক্সই" (EXE) ফাইল লক্ষ্য করেছেন। এই ইএক্সই EXE) ফাইলগুলোই হচ্ছে কম্পাইলারের শেষ ফলাফল বা আউটপুট। মানুষের বোধগম্য প্রোগ্রাম থেকে সরাসরি চালানোর মতো যন্ত্রের বোধগম্য প্রোগ্রাম রয়েছে এগুলোর মধ্যে।

জাভা প্রোগ্রামিং ভাষায় কম্পিউটার প্রোগ্রাম লেখা শুরু করতে গেলে প্রথমেই আপনার দরকার হবে জাভা ভাষার জন্য সংকলক (বা কম্পাইলার)। পরবর্তী অংশটি আপনাকে দেখাবে কিভাবে একটি সংকলক (বা কম্পাইলার) ডাউনলোড এবং ইনস্টল করতে হয়। আপনি একটি কম্পাইলার পেলেই আমরা শুরু করতে পারি। এই প্রক্রিয়ায় আমাদের কয়েক ঘন্টা লাগবে যার বেশীরভাগই হচ্ছে কিছু বড় বড় ফাইল ডাউনলোড করার সময়। আরও লাগবে ডিস্কে

৪০ মেগাবাইট ফাঁকা জায়গা (শুরু করার আগেই ফাঁকা জায়গা আছে কিনা নিশ্চিত হয়ে নিন)।

জাভা কম্পাইলার ডাউনলোড:

আপনার মেশিনে জাভা তৈরী করার পরিবেশ পেতে চাইলে - "বানানোর পরিবেশ" (ডেভেলপমেন্ট এনভাইরনমেন্ট) এর সাহায্যে আপনি কম্পিউটার প্রোগ্রাম "তৈরী" করবেন (লিখবেন) - নীচের ধাপগুলো আপনাকে সম্পন্ন করতে হবে:

জাভা বানানোর পরিবেশ (জাভা ডেভেলপমেন্ট এনভাইরনমেন্ট) (কম্পাইলার এবং অন্যান্য জিনিসপত্র) আছে এমন একটি বড় ফাইল ডাউনলোড।

জাভার জন্য দরকারী সাহায্য আছে এমন একটি ফাইল ডাউনলোড।

আপনার মেশিনে যদি উইনজিপ(WinZip)(বা একই ধরনের একটি) এরই মধ্যে না থাকে তাহলে উইনজিপ(WinZip)আছে এমন একটি বড় ফাইল ডাউনলোড এবং ইনস্টল করতে হবে।

জাভা ডেভেলপমেন্ট এনভাইরনমেন্ট ইনস্টল।

সাহায্যকারী ফাইল ইনস্টল।

এনভাইরনমেন্ট ভেরিএবল গুলো ঠিকঠাক করা।

সব কিছু পরীক্ষা করে দেখা।

শুরু করার আগেই যদি আপনি ডাউনলোড করা ফাইলগুলো রাখার জন্য আপনার temp ডিরেক্টরীতে নতুন একটি ডিরেক্টরী তৈরী করে নেন তবে ব্যাপারগুলো বেশ সহজ হবে। আমরা একে **ডাউনলোড ডিরেক্টরী**(download directory) বলে চিনব।

জাভা ডেভেলপমেন্ট এনভাইরনমেন্ট ডাউনলোড

"http://java.sun.com/j2se/1.4.2/download.html" পেইজে (ওয়েব পেইজ) চলে যান। "Download J2SE SDK" লিংকে ক্লিক করে এস.ডি.কে.(SDK) সফটওয়্যারটি ডাউনলোড করে নিন। একটি অনুমতির চুক্তিপত্র দেখা যাবে। "Accept" এ ক্লিক করুন। আপনার অপারেটিং সিস্টেম (Operating System) বেছে নিন এবং ফাইলটি আপনার ডাউনলোড ডিরেক্টরিতে ডাউনলোড করুন। বেশ বড় ফাইল এটি, সাধারণ ফোন-লাইন মোডেম এ ডাউনলোড করতে কয়েক ঘন্টা লেগে যাবে। পরের ফাইল দুটিও বেশ বড়।

২য় ধাপ - জাভার ডকুমেন্টেশন ডাউনলোড

আপনার অপারেটিং সিস্টেম (Operating System)ঠিকমত বেছে নিন। SDK 1.4.1 documentation এ ক্লিক করে ডকুমেন্টেশন ডাউনলোড করে নিন।

৩য় ধাপ - WinZip ডাউনলোড এবং ইনস্টল

আপনার কাছে যদি WinZip বা একই ধরনের অন্য কোন সফটওয়্যার এর কোন সংস্করন না থাকে তবে http://www.winzip.com/ ওয়েব পেইজ থেকে WinZip এর মূল্যায়ন(Evaluation) কপি ডাউনলোড করে নিন। ইনস্টলের জন্য যে EXE পেয়েছেন সেটা চালান। ডকুমেন্টশন ইনস্টলের জন্য কিছুক্ষন পরেই আমরা এটা ব্যবহার করব।

৪র্থ ধাপ - ডেভেলপমেন্ট কিট ইনস্টল

১ম ধাপে ডাউনলোড করা j2sdk-1_4_1-*.exe ফাইলটি চালান(রান করেন)। স্বয়ংক্রিয়ভাবে ডেভেলপমেন্ট কিট টি খুলে যাবে এবং ইনস্টল হয়ে যাবে।

৫ম ধাপ - পরিবেশ ঠিক করা

ডকুমেন্টশন ইনস্টলের দিক-নির্দেশনাগুলো পড়ে নিন। এই মাত্র ডেভেলপমেন্ট কিট টি যে ডিরেক্টরিতে ইনস্টল করেছেন সেখানেই ডকুমেন্টশন ফাইল রাখার জন্য আপনি নির্দেশ পাবেন। ডকুমেন্টশনটি আনজিপ(Unzip)করেন এবং এটা সঠিক জায়গায় চলে যাবে।

৬ষ্ঠ ধাপ - পরিবেশ ঠিক করা

এই পৃষ্ঠার নির্দেশ অনুযায়ী আপনার পাথ চলক(Path Variable) ঠিক করে নিতে হবে। MS-DOS Prompt খুলে Path বর্তমানে কোথায় দেয়া আছে দেখার জন্য PATH লিখে এটা খুব সহজে করা যায়। এরপর নোটপ্যাডে(Notepad) autoexec.bat খুলে নির্দেশনা অনুযায়ী PATH এ প্রয়োজনীয় পরিবর্তন করে নিন।

৭ম ধাপ - পরীক্ষা

আশা করি, এখন আপনি আরেকটি MS-DOS উইন্ডো খুলে **javac**লিখতে পারবেন। সবকিছু ঠিক থাকলে আপনি দুই লাইনের ছোট্ট একটি বাক্য দেখতে পাবেন। এখানে দেখতে পাবেন **javac** কিভাবে ব্যবহার করতে হয়। অর্থাৎ আপনি যাত্রা শুরু করতে পারেন। আর "Bad Command or File Name" এই ধরনের কোন বার্তা পেলে বুঝতে হবে আপনি শুরু করতে পারবেন না। ইনস্টলের নির্দেশনাগুলো আরেকবার পড়ে দেখুন কোথায় ভুল হয়েছে। PATH ঠিকমত দিয়েছেন কিনা এবং এটা কাজ করছে কিনা নিশ্চিত হয়ে নিন। পিছনে যেয়ে আবার "Programmer's Creed" অংশটি পড়ে নিন এবং সমস্যা সমাধান না হওয়া পর্যন্ত নাছোড়বান্দার মত বসে থাকেন।

এখন আপনি এমন একটি মেশিনের গর্বিত মালিক যেটা জাভা প্রোগ্রাম কম্পাইল করতে পারে। আপনি এখন সফটওয়্যার তৈরী করা শুরু করতে পারেন!

ওহ্ একটি কথা, আপনি যে জিনিসগুলো এইমাত্র আনপ্যাক করলেন তার মধ্যে একটি হচ্ছে সুন্দর-সুন্দর উদাহরনে(example) ঠাসা **demo** ডিরেক্টরী। সবগুলো উদাহরন চালানোর মত উপযোগী। তাই আপনি হয়ত

ডিরেক্টরীটা বের করে নমুনাগুলোকে একটু নেড়েচেড়ে দেখতে চাচ্ছেন। এদের অনেকগুলোই শব্দ করে। তাই অবশ্যই আপনার স্পীকার অন করে নিন। উদাহরনগুলো চালানোর জন্য example1.html এর মত দেখতে পেইজগুলো খুজে বের করুন এবং এদেরকে আপনার ওয়েব ব্রাউজারে লোড করুন।

আপনার প্রথম প্রোগ্রাম

আপনার প্রথম প্রোগ্রামটি বেশ ছোট এবং সুন্দর হবে। প্রোগ্রামটি আকাআকির জন্য একটি জায়গা তৈরী করবে এবং এর বরাবর একটি রেখা আকবে। এই প্রোগ্রামটি তৈরী করার জন্য আপনার প্রয়োজন হবে:

- প্রোগ্রামটি Notepad খুলে লিখে (অথবা কাট বা পেস্ট করে) নিন।
- প্রোগ্রামটি সেইভ করে নিন।
- জাভা এ্যাপ্লেট(Java applet) তৈরীর জন্য জাভা কম্পাইলার দিয়ে প্রোগ্রামটি কম্পাইল করুন।
- কোন সমস্যা থাকলে ঠিক করে নিন।
- একটি HTML ওয়েব পেইজ তৈরী করুন। আপনার তৈরী করা জাভা এ্যাপ্লেট এর ভিতরে থাকবে।
- জাভা এ্যাপ্লেট টি চালান।

এই পরীক্ষার জন্য আমরা যে প্রোগ্রামটি ব্যবহার করব:

```
import java.awt.Graphics;
public class FirstApplet extends java.applet.Applet
{
   public void paint(Graphics g)
   {
      g.drawLine(0, 0, 200, 200);
   }
}
```

১ম ধাপ - প্রোগ্রামটি লেখা:

আপনার প্রোগ্রামটি রাখার জন্য নতুন একটি ডিরেক্টরী তৈরী করে নিন। নোটপ্যাড (বা অন্য যে কোন টেক্সট এডিটর যেটা TXT ফাইল তৈরী করতে পারে)খুলেন। একটা ব্যাপার গুরুত্বপূর্ণ: আপনি যখন লিখবেন ছোট এবং

বড় হাতের অক্ষর আলাদা-আলাদা অর্থ বহন করবে। অর্থাৎ প্রোগ্রামে যেভাবে আছে ঠিক সেভাবেই আপনাকে ছোট এবং বড় হাতের অক্ষর লিখতে হবে। প্রোগ্রামারের মত-বিশ্বাস উপরের অংশটি আরেকবার দেখে নিন। যেভাবে আছে আপনি যদি অক্ষরে অক্ষরে সেভাবে না লিখেন তাহলে কিন্তু এটা কাজে আসবে না।

২য় ধাপ - ফাইলটি সেইভ করা:

১ম ধাপে তৈরী করা ডিরেক্টরীতে **FirstApplet.java** নামে ফাইলটি সেইভ করুন। ফাইলের নামের ক্ষেত্রে ছোট বা বড় হাতের অক্ষর ভিন্ন অর্থ বহন করে। নিশ্চিত হয়ে নিন, যেভাবে দেখানো হয়েছে, 'F' এবং 'A' বড় হাতের এবং অন্য অক্ষরগুলো ছোট হাতের।

৩য় ধাপ - প্রোগ্রামটি কম্পাইল করা:

একটি MS-DOS উইন্ডো খুলেন। ডিরেক্টরী পরিবর্তন ("cd") করে **FirstApplet.java** যেখানে আছে সেই ডিরেক্টরীতে চলে যান।

```
javac FirstApplet.java
```

ছোট এবং বড় হাতের অক্ষর ভিন্ন অর্থ বহন করে! হয় এটা কাজ করবে, যেখানে উইন্ডোতে কোন লেখা দেখা যাবে না অথবা কিছু সমস্যা থাকবে। কোন সমস্যা না থাকলে, FirstApplet.java ফাইলের পড়েই FirstApplet.class নামে একটি ফাইল তৈরী হবে।

(নিশ্চিত হয়ে নিন, **FirstApplet.java.txt** নামে সেইভ না করে **FirstApplet.java** নামে ফাইলটি সেইভ করেছেন। MS-DOS উইন্ডোতে **dir** লিখে ফাইলের নামের দিকে লক্ষ্য করলেই খুব সহজে বের করতে পারবেন। ফাইলের নামে যদি **.txt** এক্সটেনশন থাকে, নাম পরিবর্তন করে এটা বাদ দিয়ে দিন। অথবা, উইন্ডোজ এক্সপ্লোরার চালিয়ে View মেনু থেকে Options বেছে নিন। নিশ্চিত হয়ে নিন যে, "Hide MS-DOS File Extensions for file types that are registered" বক্সটি চেক্ড করা নেই এবং তারপরএক্সপ্লোরার দিয়ে ফাইলের নাম দেখেন। দরকার হলে পরিবর্তন করে নিন।)

৪র্থ ধাপ - কোন সমস্যা থাকলে ঠিক করা:

কোন সমস্যা থাকলে ভালো করে দেখে নিন। আপনার প্রোগ্রামটির সাথে উপরের প্রোগ্রামটির তুলনা করেন এবং তাদের হুবহু মিলিয়ে নিন। যতক্ষন আর কোন ভূল দেখবেন না বারবার কম্পাইল করতে থাকেন। যদি মনে হয় **javac** কাজ করছে তবে আগের অনুচ্ছেদ দেখে আপনার ব্যবস্থাটি ঠিক করে নিন।

৫ম ধাপ - একটি পেইজ তৈরী করা:

এ্যাপ্লেটটি রাখার জন্য একটি HTML পেইজ তৈরী করে নিন। আরেকটি নোটপ্যাড উইন্ডো খুলে নীচের অংশটি লিখুন:

```
<html>
<body>
<applet code=FirstApplet.class width=200 height=200>
</applet>
```

```
</body>
</html>
```

ঐ একই ডিরেক্টরীতে ফাইলটি **applet.htm** নামে সেইভ করেন।

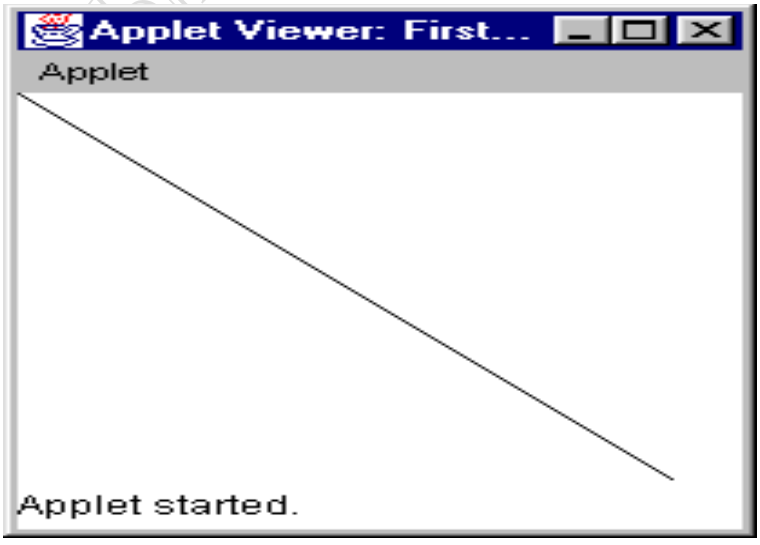
[HTML এ কাজ করার কোন পূর্ব অভিজ্ঞতা যদি আপনার না থাকে তাহলে পড়ে নিন ওয়েব পেইজ কিভাবে কাজ করে। একটি ওয়েব পেইজ এর ভিতরে জাভা এ্যাপ্লেটকে কিভাবে ব্যবহার করবেন সেটা Applet ট্যাগ দিয়ে বোঝানো হয়।]

৬ষ্ঠ ধাপ - এ্যাপ্লেটটি চালানো

আপনার MS-DOS উইন্ডোতে লিখুন:

```
appletviewer applet.htm
```

আশা করি, উপরের বাম দিকের কোনা থেকে নীচের ডান দিকে কোনা পর্যন্ত আড়াআড়ি একটা রেখা দেখতে পাবেন:



রেখাটি সম্পূর্ণ দেখার জন্য এ্যাপ্লেট ভিউয়ারকে টেনে একটু বড় করে নিন।
আপনি এই HTML পেইজকে যেকোন আধুনিক ব্রাউজার যেমন
নেটস্কেপ নেভিগেটর বা মাইক্রোসফট ইন্টারনেট এক্সপ্লোরারে স্থাপন
করে হুবহু প্রায় একই চিত্র দেখতে পারবেন।
আপনি আপনার প্রথম প্রোগ্রাম সাফল্যের সাথে তৈরী করে ফেললেন!!!

যা হয়ে গেল বুঝা:

এইমাত্র কি হলো? প্রথমে আপনি নিতান্তই সাধারন একটি **জাভা এ্যাপ্লেট** এর জন্য এক টুকরো কোড লিখলেন। এ্যাপ্লেট হচ্ছে একটি জাভা প্রোগ্রাম যেটা একটি ওয়েব ব্রাউজারে চলতে পারবে। অন্যদিকে, **জাভা এ্যাপ্লিকেশন** হচ্ছে একটি একক প্রোগ্রাম যেটা চলবে আপনার লোকাল ম্যাশিনে (জাভা এ্যাপ্লিকেশন একটু বেশী জটিল এবং মোটামুটি কম জনপ্রিয় হয়, তাই এ্যাপ্লেট দিয়ে আমরা শুরু করব)। **javac** দিয়ে আমর এ্যাপ্লেট কম্পাইল করেছি। তারপর এ্যাপ্লেটটি ধারন করার জন্য আমরা তৈরী করলাম অতি সাধারন ওয়েব পেইজ। **appletviewer** দিয়ে আমরা এ্যাপ্লেটটি চালিয়েছি, কিন্তু একটি ব্রাউজারে আপনি খুবই সহজেই চালাতে পারবেন।
প্রোগ্রামটির নিজের ১০টি লাইন রয়েছে:


```
import java.awt.Graphics;

public class FirstApplet extends java.applet.Applet
{
   public void paint(Graphics g)
```

```
    {
        g.drawLine(0, 0, 200, 200);
    }
}
```

আপনি যতগুলো জাভা এ্যাপ্লেট তৈরী করতে পারবেন তার মধ্যে এটি হচ্ছে সব থেকে সহজের একটি। প্রোগ্রামটি ভালোভাবে বুঝতে হলে আপনাকে বেশ ভালোই শিখতে হবে, বিশেষ করে **অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং টেকনিকস্** এর ক্ষেত্রগুলো। যেহেতু আমি ধরেই নিচ্ছি প্রোগ্রমিং এর অভিজ্ঞতা আপনার একেবারেই শূণ্য, এই প্রোগ্রামের ভিতরে আপতত একটি লাইনে আপনার দৃষ্টি আকর্ষন করতে চাচ্ছি:

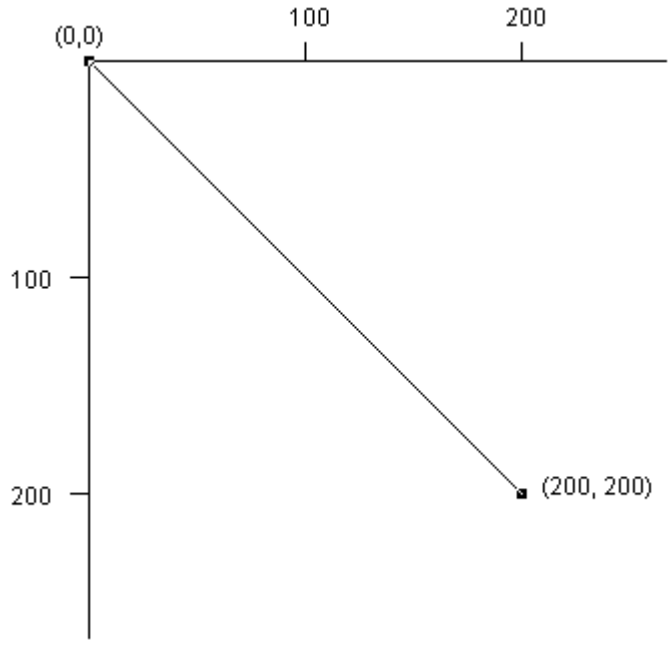
```
g.drawLine(0, 0, 200, 200);
```

প্রোগ্রামের ভিতরে আসল কাজটি করে এই লাইন। এটা আড়াআড়ি রেখাটি আঁকে। প্রোগ্রামের বাকী অংশ ঐ একটি লাইনকে সহায়তা দেয়ার জন্য মঞ্চ তৈরী করে এবং আপাতত আমরা ঐ মঞ্চের কথা বাদ দিতে পারি। আসলে যা ঘটল তা হচ্ছে, কম্পিউটারকে আমরা উপরের বাম কোনা(0,0) থেকে নীচের ডান দিকের কোনা(200, 200) পর্যন্ত একটি রেখা আঁকার জন্য বলেছি। কম্পিউটার, আমরা ঠিক যেমন বলেছি, রেখাটি এঁকেছে। এটাই হচ্ছে কম্পিউটার প্রোগ্রামিং এর আসল মজা।

(খেয়াল করুন, উপরে ৫ম ধাপে আমরা HTML পেইজের ভিতরে এ্যাপ্লেটের উইন্ডোর মাপ দিয়েছি দৈর্ঘ্য 200 এবং প্রস্থ 200।)

এই প্রোগ্রামে আমরা একটি **drawLine** নামে একটি **method** (a.k.a. **function**) কল করেছি এবং একে চারটি প্যারামিটার (**parameters**) (0,0,200,200) পাঠিয়েছি। লাইনটি একটি সেমিকোলনে যেয়ে শেষ হয়। সেমিকোলন লাইনের শেষে দাড়ির মত কাজ করে। লাইনটি শুরু হয় **g.** দিয়ে, অর্থাৎ আমরা **g** নামের নির্দিষ্ট অবজেক্ট (যেটা, এক লাইন উপরে আপনি দেখবেন, হল **Graphics** ক্লাসের - পরে এই অনুচ্ছেদে বিস্তারিতভাবে ক্লাস এবং ক্লাসের মেথড্

এর ভিতরে আমরা যাব) এর কোন মেথডকে কল করতে চাচ্ছি। মেথড্ হচ্ছে শুধুমাত্র একটি নির্দেশ - এটা কম্পিউটারকে কিছু করতে বলে। এই ক্ষেত্রে, drawLine কম্পিউটারকে (0, 0) ও (200, 200) বিন্দুর মধ্যে একটি রেখা টানতে বলে। ধরতে পারেন, উইন্ডোর 0,0 বিন্দু রয়েছে উপরের বাম কোনায় এবং X ও Y অক্ষের ধনাত্মক দিক ডানে ও নীচে বাড়তে থাকে। পর্দার উপর প্রতিটি বিন্দু (প্রতিটি pixel)হচ্ছে স্কেলের একক বৃদ্ধি।



ঐ চারটি প্যারামিটারের জায়গায় বিভিন্ন সংখ্যা বসিয়ে পরীক্ষা করার চেষ্টা করুন। একটা বা দুটা সংখ্যা পরিবর্তন করুন, পরিবর্তনগুলো সেইভ করুন, javac দিয়ে আবার কম্পাইল করুন এবং প্রতি পরিবর্তনের পর appletviewer এ পুনরায় রান করুন আর দেখুন আপনার আবিষ্কার।

drawLine ছাড়া আর কি কি ফাংশন রয়েছে? Graphics ক্লাসের ডকুমেন্টশন ঘাঁটলেই আপনি পেয়ে যাবেন। আপনি যখন জাভা

ডেভেলপমেন্টম কিট ইনস্টল করেন এবং ডকুমেন্টশন আনপ্যাক্ করেন, একটা ফাইল আনলোড হয় এর মধ্যেই আর এটার নাম হচ্ছে **java.awt.Graphics.html**, এবং এটি আপনার মেশিনেই আছে। এই ফাইলটাই **Graphics** ক্লাস সম্বন্ধে বিস্তারিত লিখেছে। আমার মেশিনে এই ফাইলের সঠিক পাথ(Path)টা হচ্ছে D:\jdk1.1.7\docs\api\java.awt.Graphics.html. আপনার মেশিনে পাথটা একটু আলাদা হতে পারে, কিন্তু খুব কাছাকাছি - এটা নির্ভর করছে আপনি ঠিক কোথায় ইনস্টল করেছেন। ফাইলটি খুঁজে বের করে খুলুন। ফাইলটির একদম উপরের দিকে একটি অংশ হচ্ছে "Method Index"। এটি হচ্ছে এই ক্লাসে যতগুলো মেথড রয়েছে তার একটি তালিকা। **drawLine** মেথড হচ্ছে তাদের একটি, কিন্তু আপনি আরও অনেক দেখতে পাবেন সেখানে। অন্যান্য বিষয়ের পাশাপাশি আপনি আঁকতে পারেন:

- Lines
- Arcs
- Ovals
- Polygons
- Rectangles
- Strings
- Characters

এইসব বিভিন্ন মেথড্গুলো একটু পড়ে দেখুন এবং এগুলো নিয়ে পরীক্ষা করার চেষ্টা করুন। দেখুন নতুন কি তৈরী করা সম্ভব। উদাহরণ হিসেবে এই কোডটুকু দেখতে পারেন:

```
g.drawLine(0, 0, 200, 200);
```

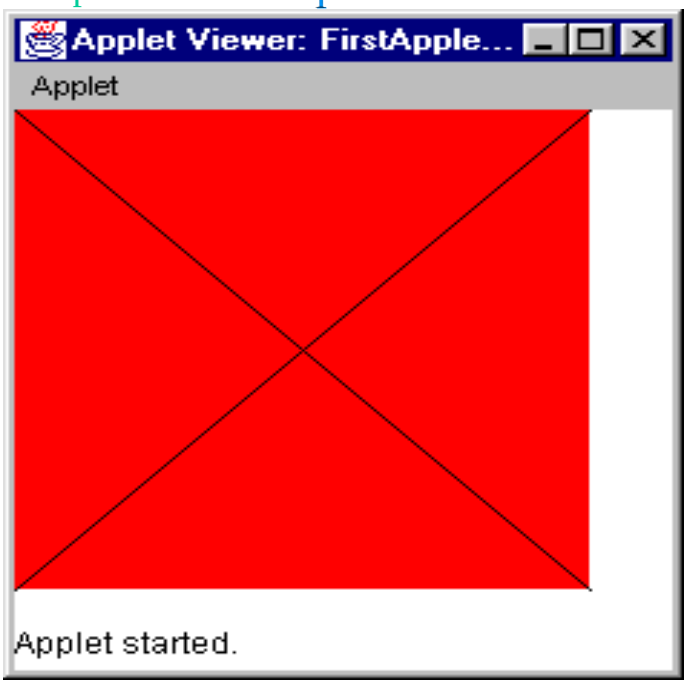
```
        g.drawRect(0, 0, 200, 200);
        g.drawLine(200, 0, 0, 200);
```

এতে একটা দুই কর্ণবিশিষ্ট বাক্স দেখতে পাবেন(ছবিটি পুরোপুরি দেখার জন্য উইন্ডোকে প্রয়োজনমত বড় করে নিতে ভুলবেন না)।

```
import java.awt.Graphics;
import java.awt.Color;

public class FirstApplet extends java.applet.Applet
{
    public void paint(Graphics g)
    {
        g.setColor(Color.red);
        g.fillRect(0, 0, 200, 200);
        g.setColor(Color.black);
        g.drawLine(0, 0, 200, 200);
        g.drawLine(200, 0, 0, 200);
    }
}
```

লক্ষ্য করুন, প্রোগ্রামের দ্বিতীয় লাইনে নতুন একটি import লাইন যোগ হয়েছে। প্রোগ্রামটির আউটপুট দেখতে হবে:



আপনার মাথায় হয়ত একটা চিন্তা চলছে যে, "শুধুমাত্র red বাদ দিয়ে Color.red এর ব্যবহার জানল কিভাবে এবং দ্বিতীয় import লাইন যোগ করতেই বা জানল কিভাবে?" এইসব ব্যাপারগুলো উদাহরণ দিয়ে শিখবেন। কারণ, শুধুমাত্র setColor মেথড্ কল করার একটা উদাহরণ দেখেছেন আপনি। আপনি এখন জানেন যে, রং পরিবর্তন করতে চাইলে Color. এর পরে একটি রং এর নাম বসিয়ে setColor মেথড্ এর প্যারামিটার হিসেবে ব্যবহার করতে হয় এবং প্রোগ্রামের একদম উপরে সঠিক import লাইন যোগ করতে হয়। আপনি যদি setColor এর দিকে তাকান দেখতে পাবেন এর একটি সংযুক্ত অংশ রয়েছে Color ক্লাস সম্বন্ধে। এবং এর মধ্যে রয়েছে সকল সঠিক রং এর নামযুক্ত একটি তালিকা এবং নতুন রং (নামবিহীন)তৈরীর উপায়। আপনি তথ্যটি পড়েছেন, আপনার মাথায় রেখে দিয়েছেন এবং এখন আপনি জানেন জাভাতে কিভাবে রং পরিবর্তন করা যায়। কম্পিউটার প্রোগ্রামার হবার এটাই হচ্ছে সবচেয়ে গুরুত্বপূর্ণ অংশ - আপনি কৌশল শিখবেন এবং পড়ের প্রোগ্রাম লিখার জন্য তা মনে রাখবেন। হয় উদাহরণ পড়ে (এখানে যেমন করেছেন) নতুবা ডকুমেন্টেশন পড়ে বা কোডের নমুনা

(যেমন demo ডিরেক্টরীর মধ্যে) দেখে আপনি কৌশল শিখবেন।
অনুসন্ধান করার মত, নতুন জিনিস শিখার মত এবং বিভিন্ন জিনিস মনে রাখার মত একটা ব্রেইন থাকলে তবেই আপনি প্রোগ্রামিং কে ভালোবাসতে পারবেন!
এই অংশে আপনি যা শিখলেন: লিনিয়ার, সিকুয়েন্সাল কোড - উপর থেকে শুরু করে নীচ পর্যন্ত মেথড কল সম্বলিত ভাগ-ভাগ করা কোড লিখা(লাল চতুর্ভূজ আঁকার আগে লাইনগুলোর একটি আঁকার চেষ্টা করে দেখুন কি হয় - এটা চতুর্ভূজটি দ্বারা ঢেঁকে যাবে এবং অদৃশ্য হয়ে যাবে। কোডের ভিতরে লাইনগুলোর ক্রম একটি গুরুত্বপূর্ণ ব্যাপার)। কোডের ধারাবাহিক লাইন একটি কম্পিউটার প্রোগ্রামের মূল অংশ গঠন করে। আঁকার সব ধরনের ভিন্ন-ভিন্ন মেথড্ নিয়ে পরীক্ষা-নিরীক্ষা করে দেখুন নতুন কিছু তৈরী করতে পারেন কিনা।

বাগস এবং ডিবাগিং(Bugs and Debugging):

আপনি প্রোগ্রামিং শিখছেন। আর তাই একটা ব্যাপার আপনি লক্ষ্য করবেন যে, আপনি ক্রমেই বেশ ভালই ভূল ও অনুমান করছেন। এর কারনে আপনার প্রোগ্রাম হয় কম্পাইল হচ্ছে না বা এক্সিকিউট করার পর আপনি আশানুরুপ আউটপুট পাচ্ছেন না। এই সমস্যাগুলোই হচ্ছে বাগ্ বা ত্রুটি। আর এদের দূর করার প্রক্রিয়াকে বলা হয় ডিবাগিং। যে কোন প্রোগ্রামারের প্রায় অর্ধেক সময় ব্যয় হয় এই ডিবাগিং এ।
আপনার নিজের বাগ্ তৈরী করার জন্য আপনি প্রচুর সময় এবং সুযোগ পাবেন। কিন্তু তার আগে বাগ্ আছে কি না এই ব্যাপারে আরও বেশী পরিষ্কার হবার জন্য নিজেরাই কিছু তৈরী করি। আপনার প্রোগ্রামের ভিতরে কোন একটা লাইনের শেষে সেমিকোলন বাদ দিয়ে javac দিয়ে কম্পাইল করার চেষ্টা করেন। কম্পাইলার আপনাকে একটা ভূলের মেসেজ দিবে। এটাকে বলে **কম্পাইলার এরর** এবং প্রোগ্রাম এক্সিকিউট করার আগে আপনাকে এই ধরনের সকল এরর বা ভূল দূর করতে হবে ।
বিভিন্ন ধরনের কম্পাইলার এররের সাথে পরিচিত হতে ফাংশনের ভূল বানান দিয়ে, একটা "{" বাদ দিয়ে বা কোন একটা ইমপোর্ট লাইন বাদ দিয়ে দেখতে পারেন। প্রথমেই কোন এক বিশেষ ধরনের কম্পাইলার এরর দেখে হতাশ হতে পারেন। কিন্তু বিশেষ উদ্দেশ্যে তৈরী পরিচিত

এরর দিয়ে এভাবে পরীক্ষা-নিরীক্ষা করতে করতে আপনি অনেক সাধারন এরর চিনতে পারবেন।

প্রোগ্রাম যখন কম্পাইল হতে থাকে এবং রান হতে থাকে তখন এক্সিকিউশন (বা রান-টাইম) এরর নামে পরিচিত বাগ দেখা যেতে পারে। এবং এতে আপনি পূর্ব পরিকল্পিত আউটপুট পাবেন না। উদাহরণস্বরুপ বলা যায়, নীচের কোডটি দু'টি পরস্পরছেদী কর্ণবিশিষ্ট একটি চতুর্ভূজ আঁকে:

```
g.setColor(Color.red);
g.fillRect(0, 0, 200, 200);
g.setColor(Color.black);
g.drawLine(0, 0, 200, 200);
g.drawLine(200, 0, 0, 200);
```

অন্যদিকে নীচের কোডটি শুধুমাত্র লাল চতুর্ভূজটিকে(লাইন দু'টির উপর দিয়ে) তৈরী করে:

```
g.setColor(Color.black);
g.drawLine(0, 0, 200, 200);
g.drawLine(200, 0, 0, 200);
g.setColor(Color.red);
g.fillRect(0, 0, 200, 200);
```

এই কোডটি প্রায় হুবহু একই রকম কিন্তু যখন এক্সিকিউট হয় সম্পূর্ণ উল্টো দেখা যায়। আপনি যদি কর্ণ দু'টি দেখতে চান তবে দ্বিতীয় কোডটিতে একটি বাগ রয়েছে।

আরেকটি উদাহরণ দেয়া হল:

```
g.drawLine(0, 0, 200, 200);
g.drawRect(0, 0, 200, 200);
g.drawLine(200, 0, 0, 200);
```

এই কোড তৈরী করবে চারদিকে কালো লাইন বিশিষ্ট একটি চতুর্ভূজ এবং দুটি কর্ণ। পরের কোডটি এখানে তৈরী করবে শুধুমাত্র একটি কর্ণ:

```
g.drawLine(0, 0, 200, 200);
```

```
g.drawRect(0, 0, 200, 200);
g.drawLine(0, 200, 0, 200);
```

আপনি যদি দু'টি কর্ণ দেখার আশা করে থাকেন তবে আবার দ্বিতীয় কোডে একটি বাগ রয়ে গেছে(কি ভূল ছিল বুঝতে না পারা পর্যন্ত ২য় কোডটুকু দেখতে থাকেন)। এই ধরনের বাগ খুজে পেতে অনেক সময় লাগে কারণ এগুলো খুব সূক্ষ্ম।

নিজের বাগ ধরা চর্চা করার জন্য আপনি পর্যাপ্ত সময় পাবেন। গড়পড়তা একজন প্রোগ্রামার তার প্রায় অর্ধেক সময় বাগ খুজে বের করা এবং দূর করায় ব্যয় করেন। যখন বাগ দেখা যাবে দয়া করে নিরাশ হবেন না - এগুলো প্রোগ্রামিং জগতের খুব সাধারণ একটা অংশ।

ভেরিয়েবলস

সাময়িকভাবে ডাটা বা তথ্য রাখার জন্য সকল প্রোগ্রাম ভেরিয়েবল ব্যবহার করে। উদাহরনস্বরূপ ধরুন প্রোগ্রামের কোন এক পর্যায়ে আপনি একজন ইউজারের কাছ থেকে একটি সংখ্যা চাইলেন, পরে যেন ব্যবহার করতে পারেন সেজন্য সংখ্যাটিকে আপনি একটি ভেরিয়েবল এ রাখবেন।

ভেরিয়েবল ব্যবহার করার আগে অবশ্যই তাদেরকে সুনির্দিষ্ট(ডিফাইনড) বা সুস্পষ্টভাবে জানানো(ডিক্লেয়ার্ড) থাকতে হবে। এবং প্রত্যেক ভেরিয়েবলকে কোন বিশেষ ধরন(টাইপ) হিসেবে উল্লেখ করতে হবে। উদাহরনস্বরূপ, আপনি একটি ভেরিয়েবলকে এমন এক ধরন(টাইপ) হিসেবে উল্লেখ করতে পারেন যা সংখ্যা রাখতে পারে, আবার আরেকটি ভেরিয়েবল উল্লেখ করতে পারেন যা কোন ব্যক্তির নাম রাখতে পারে। (কারন জাভাতে ভেরিয়েবল ব্যবহার করার পূর্বে তাদের অবশ্যই নির্দিষ্টভাবে উল্লেখ করতে হয় এবং যে রকম তথ্য সেখানে থাকবে তাদের ধরন বা টাইপ উল্লেখ করতে হয়, জাভা হচ্ছে একটি **স্ট্রংলি টাইপড** ল্যাংগুয়েজ। অনেক ল্যাংগুয়েজের এই ধরনের বাধা ধরা নিয়ম-কানুন নেই। সাধারনত বড় বড় প্রোগ্রাম তৈরী করার সময় স্ট্রং টাইপিং আপনার প্রোগ্রামিং এরর এর সংখ্যা কমাতে সাহায্য করে।)

```
import java.awt.Graphics;
import java.awt.Color;

public class FirstApplet extends java.applet.Applet
{
   public void paint(Graphics g)
   {
      int width = 200;
      int height = 200;
      g.drawRect(0, 0, width, height);
      g.drawLine(0, 0, width, height);
      g.drawLine(width, 0, 0, height);
   }
}
```

উপরের প্রোগ্রামে আমরা দুটি ভেরিয়েবল বা চলক- উল্লেখ **width** ও **height** করেছি। এদের ধরন দিয়েছি **int**। একটি ভেরিয়েবল বা চলক যেকোন ইনটিজার(পূর্ণ সংখ্যা যেমন ১, ২, ৩)রাখতে পারে। দুটি ভেরিয়েবলকেই আমরা ২০০ দিয়ে ইনিসিয়ালাইজ করেছি। খুব সহজেই আমরা লিখতে পারতাম:

```
int width;
width = 200;
int height;
height = 200;
```

১ম পদ্ধতিতে শুধুমাত্র একটু তাড়াতাড়ি টাইপ করা যাবে।
কোন একটি ভেরিয়েবলকে তার প্রথম ভেলু বসানোর পদ্ধতিকে বলা হয় ভেরিয়েবল এর ইনিসিয়ালাইজিং। ভেরিয়েবলকে ইনিসিয়ালাইজ করতে ভূলে গেলে খুব সাধারন একটি বাগ দেখা দেয়। বাগটি দেখতে চাইলে কোডের ইনিসিয়ালাইজিং অংশটুকু("= 200" অংশটি) বাদ দিন এবং

প্রোগ্রামটি আবার কম্পাইল করে দেখুন কি হয়। আপনি দেখবেন যে কম্পাইলার সমস্যাটি সম্বন্ধে আপনাকে অভিযোগ করছে। যাই হোক এটি একটি ভালো বৈশিষ্ট্য। এটি আপনাকে অনেক সময় অপচয়ের হাত থেকে রক্ষা করবে।

জাভাতে দুই ধরনের ভেরিয়েবল আছে - সাধারন (**প্রিমিটিভ**) ভেরিয়েবল এবং **ক্লাস**।

int টাইপটি সাধারন। এটি যা করতে পারে তা হচ্ছে এটি একটি সংখ্যা রাখতে পারে। আপনি একটি **int** ঘোষনা(ডিক্লেয়ার) করবেন, এতে একটি ভেলু দিবেন এবং ব্যবহার করবেন। অন্যদিকে, ক্লাস এ কয়েক অংশ থাকে এবং কিছু মেথড থাকে। এই মেথডগুলোর সাহায্যে ক্লাসকে খুব সহজে ব্যবহার করা যায়। ক্লাসের সরাসরি একটি ভালো উদাহরন হলো **Rectangle** ক্লাস। তাই এটা দিয়েই আমরা শুরু করি।

এতক্ষন পর্যন্ত আমরা যে প্রোগ্রামটির উপর কাজ করলাম তার মধ্যে সবচেয়ে বড় একটি সীমাবদ্ধতা হচ্ছে যে, আমরা ধরে নিয়েছি উইন্ডোটি ২০০ বাই ২০০ পিক্সেলের। আমরা যদি উইন্ডোকে জিজ্ঞাসা করি "তুমি কত বড়?" এবং তারপর চতুর্ভূজ এবং কর্ণদ্বয় আঁকি তাহলে কেমন হবে?

Graphics ক্লাস (java.awt.Graphics.html - আঁকাআঁকির সকল ফাংশন যে ফাইলের মধ্যে আছে) এর ডকুমেন্টশন পেইজটি যদি আপনি দেখেন তাহলে লক্ষ্য করবেন যে, এদের মধ্যে **getClipBounds** নামে একটি ফাংশন আছে। সম্পূর্ণ বিবরণ দেখার জন্য ফাংশনটির উপরে ক্লিক করুন। ফাংশনটি কোন প্যারামিটার নেয় না কিন্তু এর পরিবর্তে **Rectangle** টাইপের একটি ভেলু রিটার্ন করে। রিটার্ন করা এই rectangle এর ভিতরে রয়েছে ড্রয়িং এরিয়ার দৈর্ঘ্য এবং প্রস্থ। ডকুমেন্টশন এর এই পেইজে **Rectangle** এর উপর ক্লিক করলে আপনি চলে যাবেন **Rectangle** ক্লাস (java.awt.Graphics.html) এর ডকুমেন্টশন পেইজে।

পেইজটির উপরে ভেরিয়েবল এর তালিকার অংশে তাকালে দেখতে পাবেন যে এই ক্লাসে x, y, width ও height নামে যথাক্রমে চারটি ভেরিয়েবল রয়েছে। সুতরাং এখন আমরা যেটা করব তা হচ্ছে **getClipBounds** ব্যবহার করে সীমানা হিসেবে একটি rectangle অবজেক্ট পাব এবং তা থেকে দৈর্ঘ্য এবং প্রস্থ বের করে **width** ও

height ভেরিয়েবলে সংরক্ষন করব। পূর্বের উদাহরনে আমরা ঠিক এভাবেই করেছি:

```
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Rectangle;

public class FirstApplet extends java.applet.Applet
{
   public void paint(Graphics g)
   {
      int width;
      int height;
      Rectangle r;

      r = g.getClipBounds();
      width = r.width - 1;
      height = r.height - 1;

      g.drawRect(0, 0, width, height);
      g.drawLine(0, 0, width, height);
      g.drawLine(width, 0, 0, height);
   }
}
```


এই উদাহরনটি রান করলে দেখতে পাবেন যে চতুর্ভূজ এবং কর্ণগুলো অঙ্কনের জায়গার সাথে সম্পূর্ণ ফিট হয়েছে। আরও দেখবেন, উইন্ডোর সাইজ পরিবর্তনের সাথে সাথে স্বয়ংক্রিয়ভাবে চতুর্ভূজ এবং কর্ণগুলো নতুন সাইজে আবার অঙ্কিত হচ্ছে। এই কোডে নতুন পাঁচটি ধারনা আছে, এক নজরে দেখে নেই:

প্রথমত, Rectangle ক্লাস ব্যবহার করেছি বলে প্রোগ্রামের তৃতীয় লাইনে **java.awt.Rectangle** ইমপোর্ট করতে হবে।

প্রোগ্রামের ভিতরে আমরা তিনটি চলক বা ভেরিয়েবল ব্যবহার করেছি।

দুটি (**width** এবং **height**)**int** টাইপের এবং একটি (**r**) Rectangle টাইপের।

অঙ্কনের পরিসর বা ড্রয়িং এরিয়ার মাপ জানার জন্য আমরা

getClipBounds ফাংশন ব্যবহার করেছি। এটি কোন প্যারামিটার নেয় না,তাই এতে আমরা কিছুই দেইনি("()")। কিন্তু এটি একটি Rectangle রিটার্ন করে। "r= g.getClipBounds();" লাইনটি লিখে আমরা আসলে এটিই বলতে চেয়েছি "রিটার্ন করা চতুর্ভূজটি দয়া করে r ভেরিয়েবলে বসিয়ে দিন।

Rectangle ক্লাসের সদস্য হিসেবে r ভেরিয়েবলটি আসলে চারটি ভেরিয়েবল নিয়ে গঠিত - x, y, width এবং height (Rectangle ক্লাসের ডকুমেন্টেশন পড়ার সময় আপনি হয়ত এই নামগুলো পড়ে থাকবেন)। এদের ব্যবহার করতে হলে "."(ডটসেপারেট (র ব্যবহার করতে হবে। সুতরাং "r.width" অংশটি হচ্ছে "r ভেরিয়েবলের ভিতরে **width** এর মান"। এই মানটি বসে যায় আমাদের স্থানীয় বা লোকাল ভেরিয়েবল **width** এর ভিতরে। উপরে ঐ জায়গাটিতে আমরা ১ বিয়োগ করেছি। বিয়োগটি বাদ দিয়ে দেখুনতো কি হয়। ১ এর বদলে ৫ বাদ দিয়েও দেখুন কি হয়।

সবশেষে ড্রয়িং ফাংশনে আমরা **width** ও **height** ব্যবহার করেছি।

এই জায়গায় সাধারণত একটি প্রশ্ন করা হয়, "**width** ও **height** নামের ভেরিয়েবলগুলো তৈরী করার আদৌ কোন দরকার ছিল কি?" উত্তর হচ্ছে "না"। ড্রয়িং ফাংশনে আমরা সরাসরি **r.width - 1** ব্যবহার করতে পারতাম। ভেরিয়েবলগুলো তৈরী করার উদ্দেশ্য হচ্ছে কোড যেন একটু সহজভাবে পড়া যায়। এবং একই কারণে এইভাবে কোড লেখা একটি ভালো অভ্যাসও বটে।

জাভা কয়েকটি সহজ ধরনের ভেরিয়েবল সাপোর্ট করে। এদের মধ্যে

নীচের তিনটি হচ্ছে সবচেয়ে কমন:

- **int** - পূর্ণ সংখ্যা (১, ২, ৩ ....)
-
- **float** - দশমিক সংখ্যা (যেমন: ৩.১৪১৫৯)
-
- **char** - অক্ষর (a,b,c .....)

সাধারণ ভেরিয়েবলের উপর আপনি গাণিতিক পদ্ধতি প্রয়োগ করতে পারেন। +(যোগ), -(বিয়োগ), *(গুণ), /(ভাগ) এবং আরও কিছু আছে জাভায়। কিভাবে এই পদ্ধতিগুলো প্রোগ্রামে প্রয়োগ করবেন সে ব্যাপারে এখানে একটি উদাহরণ দেয়া হলো। ধরা যাক আপনি ১০ ফিট ব্যাস বিশিষ্ট একটি গোলকের আয়তন নির্ণয় করতে চান। নীচের কোডটি তা করে দেবে:

```
float diameter = 10;
float radius;
float volume;

radius = diameter / 2.0;
volume = 4.0 / 3.0 * 3.14159 * radius * radius * radius;
```

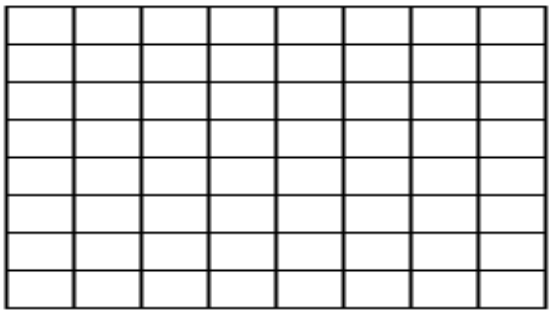
প্রথম হিসাবটি হচ্ছে "**diameter** ভেরিয়েবলের মানকে ২.০ দিয়ে ভাগ করে ভাগফলকে **radius** নামক ভেরিয়েবলে রাখো"। আপনি খেয়াল করবেন, "=" চিহ্নটি এখানে নির্দেশ করে "ডান দিকের হিসাবের ফলাফলকে বাম দিকের ভেরিয়েবলে রাখো।

লুপিং(Looping)

কম্পিউটার সুচারুভাবে করে এমন কাজগুলোর মধ্যে একটি হচ্ছে পুনরাবৃত্তিক ভাবে হিসাব - কাজ করা। পূর্বের অনুচ্ছেদগুলোর মধ্যেই দেখেছি কিভাবে "অনুক্রমিক(বা সিকুয়েন্সিয়াল) কোড" লিখতে হয়।

এখন আমরা যে বিষয়ে আলোচনা করব তা হচ্ছে কিভাবে এই অনুক্রমিক কোডের অংশগুলোকে বারবার কাজ করানো যায়।

উদাহরণরুপে ধরুন নীচের ছবিটি আপনাকে আঁকতে বলা হলো:



শুরু করার ভালো একটি স্থান হবে যদি আমরা নীচের মত অনুভূমিক সমান্তরাল রেখাগুলো এঁকে ফেলি:

লাইনগুলো আঁকার একটি উপায় হচ্ছে অনুক্রমিক(বা সিকুয়েন্সিয়াল)

কোড লেখা:

```
import java.awt.Graphics;

public class FirstApplet extends java.applet.Applet
{
   public void paint(Graphics g)
```

```
    {
        int y;
        y = 10;
        g.drawLine(10, y, 210, y);
        y = y + 25;
        g.drawLine(10, y, 210, y);
        y = y + 25;
        g.drawLine(10, y, 210, y);
        y = y + 25;
        g.drawLine(10, y, 210, y);
        y = y + 25;
        g.drawLine(10, y, 210, y);
        y = y + 25;
        g.drawLine(10, y, 210, y);
        y = y + 25;
        g.drawLine(10, y, 210, y);
        y = y + 25;
        g.drawLine(10, y, 210, y);
        y = y + 25;
        g.drawLine(10, y, 210, y);
    }
}
```


(কিছু নতুন প্রোগ্রামারের কাছে "y=y+25" অংশটি খারাপ দেখায় যখন তারা এটি প্রথমবারের মত দেখে। এর প্রকৃত অর্থ হচ্ছে, "y ভেরিয়েবলের মানের সাথে ২৫ যোগ করে যোগফলকে আবার y ভেরিয়েবলে বসানো"। সুতরাং লাইনটি এক্সিকিউট হবার আগে যদি y এর মান ১০ থাকে তাহলে লাইনটি এক্সিকিউট হবার পরেই এর মান হয়ে যাবে ৩৫।)

প্রায় সবাই কোডের দিকে তাকানোর পরপরই খেয়াল করে থাকবেন যে এর ভিতরে দুটি লাইন রয়েছে যা ঘুরেফিরে বারবার আসে। এই নির্দিষ্ট ক্ষেত্রে এই পুণরাবৃত্তি খুব একটা খারাপ হয়নি, কিন্তু চিন্তা করে

দেখুন তো আপনি যখন হাজার হাজার রো এবং কলাম সম্পন্ন একটি গ্রিড আঁকবেন প্রোগ্রাম তৈরীর এই পদ্ধতি বেশ বিরক্তিকর হয়ে দেখা দিবে। এই সমস্যার সমাধান হচ্ছে একটি লুপ, নীচে যেমন দেখানো হয়েছে:

```
import java.awt.Graphics;
public class FirstApplet extends java.applet.Applet
{
   public void paint(Graphics g)
   {
      int y;
      y = 10;
      while (y <= 210)
      {
         g.drawLine(10, y, 210, y);
         y = y + 25;
      }
   }
}
```

প্রোগ্রামটি রান করলে দেখতে পাবেন ২০০ পিক্সেল দৈর্ঘ্যবিশিষ্ট নয়টি অনুভূমিক সমান্তরাল রেখা আঁকা হয়েছে।

while যুক্ত অংশটিকে জাভাতে লুপিং অংশ বলে। ঐ অংশটি জাভাকে ঠিক এই ভাবে কাজ করতে বলে: **while** যুক্ত বাক্যংশের বন্ধনীর ভিতেরের অংশকে জাভা ভালোভাবে দেখে এবং ঠিক করে, "y কি ২১০ এর চেয়ে ছোট না সমান?"

- উত্তর যদি হ্যা হয় তাহলে জাভা "{" এবং "}" ব্রাকেটের ভিতরে

- আবদ্ধ কোডের ভিতরে ঢুকে যায়। লুপিং শেষ হয় ব্লকের একেবারে শেষে। জাভা যখন শেষের ব্রাকেটের নিকট পৌঁছে এটি আবার **while** অংশে ফিরে আসে এবং একই প্রশ্ন আবার করে। এই ঘুরে আসার ব্যাপার অর্থাৎ লুপিং এর ঘটনা অনেকবার হতে পারে।
- আর যদি উত্তর "না" হয় তাহলে জাভা ২য় ব্রাকেট দ্বারা আবদ্ধ অংশ বাদ দিয়ে স্বাভাবিকভাবে নীচের দিকে যেতে থাকবে।

সুতরাং আপনি লক্ষ করে থাকবেন যে প্রোগ্রামটি যখন আপনি রান করবেন, শুরুতে **y** এর মান থাকে ১০। ২১০ এর চেয়ে ১০ ছোট আর , তাই জাভা ব্রাকেট দ্বারা আবদ্ধ অংশে ঢুকে যায় এবং (১০, ১০) বিন্দু থেকে (২১০, ১০) বিন্দু পর্যন্ত একটি রেখা আঁকে, **y** এর মান ৩৫ করে এবং আবার **while** এর কাছে ফিরে যায়। ২১০ এর চেয়ে ৩৫ ছোট, তাই জাভা আবার ব্রাকেট এর ভিতরের অংশে ঢুকে যায়, (১০, ৩৫) থেকে (২১০, ৩৫) বিন্দু পর্যন্ত একটি রেখা আঁকে, **y** এর মান ৬০ করে এবং তারপর আবার **while** এর কাছে ফিরে যায়। এই ধারাবাহিকতা চলতে থাকে যতক্ষন না পর্যন্ত **y** এর মান ২১০ এর চেয়ে বড় না হয়। তখন প্রোগ্রাম শেষ হয়ে যায়।

আমরা দ্বিতীয় একটি লুপ যোগ করে আমাদের গ্রিড শেষ করতে পারি, ঠিক এইভাবে:

```
import java.awt.Graphics;
public class FirstApplet extends java.applet.Applet
{
   public void paint(Graphics g)
   {
```

```
        int x, y;
        y = 10;
        while (y <= 210)
        {
           g.drawLine(10, y, 210, y);
           y = y + 25;
        }
        x = 10;
        while (x <= 210)
        {
           g.drawLine(x, 10, x, 210);
           x = x + 25;
        }
    }
}
```

আপনি লক্ষ্য করে দেখবেন যে একটি **while** এর তিনটি অংশ থাকে:

- একটি আছে **ইনিশিয়ালাইজিং স্টেপ** যা **y** এর মান ১০ করে।
-
- তারপর **while** এর প্রথম ব্রাকেটের ভিতরে **এভালুয়েশন স্টেপ**।
-
- তারপর **while** এর কোন একটি জায়গায় রয়েছে **ইনক্রিমেন্ট স্টেপ**

-

যা **y** এর বৃদ্ধি করে।

একই কাজ করার জন্য জাভাতে আরেকটি পদ্ধতি আছে, কিন্তু এটি **while** এর চেয়ে একটু জটিল। এটাকে **for** লুপ বলে। আপনার কাছে নীচের মত একটি **while** অংশ থাকে:

```
y = 10;
        while (y <= 210)
        {
           g.drawLine(10, y, 210, y);
```

```
        y = y + 25;
    }
```

তাহলে এই **while** অংশের সমমানের কোড হবে ঠিক এইরকম:

```
for (y = 10; y <= 210; y = y + 25)
    {
        g.drawLine(10, y, 210, y);
    }
```

আপনি দেখবেন যে **for** যা করে তা হল এটি ইনিসিয়ালাইজিং, এভালুয়েশন এবং ইনক্রিমেন্টিং লাইনগুলোকে একটি ছোট এক লাইনে নিয়ে আসে। এটি আপনার প্রোগ্রামের দৈর্ঘ্যকে শুধুমাত্র ছোট করে, আর কিছুই না।

এর ভিতরেই লুপ সম্বন্ধে দ্রুত দুটি পয়েন্ট:

- অনেক ক্ষেত্রে, এটা লুপের ভিতরে প্রতিবার **y** এর মান ২১০ এ

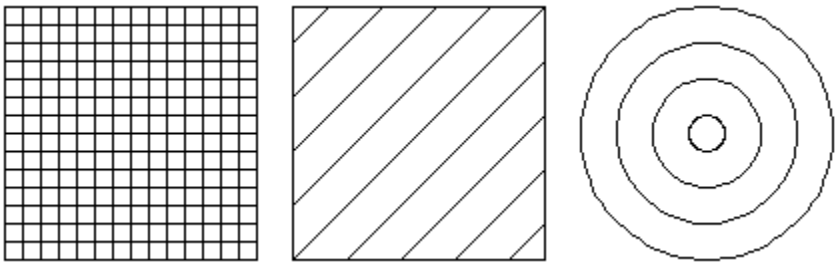
- ইনিসিয়ালাইজ করে আবার এর থেকে ২৫ কমানোর মত সোজা। এভালুয়েসন প্রশ্ন করবে, "**y** কি ১০ এর চেয়ে বড় না সমান?" পছন্দ আপনার। অনেকে প্রথমে যোগ করে তারপর বিয়োগ করা সহজ মনে করে, কিন্তু আপনি তো অন্যরকমও হতে পারেন।

- ইনক্রিমেন্ট বা মান বাড়ানোর ধাপটি বেশ গুরুত্বপূর্ণ। ধরুন ঘটনাক্রমে

- আপনি লুপের ভিতরে "y = y + 25;" শব্দগুলো ভূলে লিখলেন না। এতে করে ফলাফল যা হবে তা হচ্ছে **y** এর মান আর পরিবর্তন হবে না - এটার মান সবসময় ১০ থাকবে। অর্থাৎ **y** এর মান কখোনই ২১০ এর চেয়ে বড় হবে না এবং লুপটি সারাজীবন ধরে চলতে থাকবে (অথবা যতক্ষন না পর্যন্ত আপনি কম্পিউটার বন্ধ করবেন বা উইন্ডোটি

বন্ধ করে দিবেন)। এই ধরনের অবস্থাকে বলে **অনন্ত লুপ** বা **ইনফিনিটিভ লুপ** বলে। এই ধরনের ত্রুটি হরহামেশাই ঘটে।

লুপিং নিয়ে পরীক্ষা-নিরীক্ষা করতে চাইলে নীচের ছবিগুলো আঁকার জন্য প্রোগ্রাম লিখতে পারেন:



জাভা এবং অন্যান্য কম্পিউটার প্রোগ্রামিং ল্যাংগুয়েজ সম্বন্ধে আরও অনেক তথ্য পেতে চাইলে পরের পৃষ্ঠার লিংকগুলো দেখে নিন।

[বি:দ্র: এই অংশটি howstuffworks.com থেকে অনুবাদ করা হয়েছে]

জাভা প্রোগ্রামিং লেনগুয়েজ হাতেখড়ি- প্রথম পর্ব

অভিজ্ঞ, অনভিজ্ঞ প্রোগ্রামার এবং সাধারন ইউজার সবার জাভা প্রোগ্রামিং লেনগুয়েজ সম্বন্ধে সম্যক ধারনা দেওয়ার জন্য আমার এই ক্ষুদ্র প্রচেষ্টা।আমি একজন প্রোগ্রামার হিসাবে চেষ্টা করবো পরিচ্ছন বাংলা এবং প্রয়োজনীয় ইংরেজী ভাষার ব্যবহারসহ ছোট ছোট টপিকস আকারে ধারাবাহিকভাবে প্রত্যকটা বিষয় আলোচনা করার,আমি আশা করি জাভা প্রোগ্রামিং লেনগুয়েজ হাতেখড়ি দ্বারা সবাই উপকৃত হবেন।

প্রোগ্রাম লেনগুয়েজ এর প্রাথমিক ধারনা: কম্পিউটার আবিষ্কারের পর থেকেই তা দিয়ে জটিল সমস্যা সহজে সমাধানের চেষ্টা অব্যাহত রয়েছে এবং গবেষনা ও উন্নয়নমূলক কাজ চলছে বিভিন্ন প্রোগ্রামিং পদ্বতি নিয়ে। মডিউলার প্রোগ্রামিং, টপ-ডাউন প্রোগ্রামিং, বটম-আপ প্রোগ্রামিং উল্লখযোগ্য কয়েকটি প্রোগ্রামিং পদ্বতি। সকল প্রোগ্রামিং পদ্বতির উদ্দেশ্য অভিন্ন- সহজে সমস্যা সমাধানের জন্য প্রোগ্রাম তৈরী, প্রোগ্রামের গ্রহনযোগ্যতা বৃদ্বি, ব্যবহৃত ডাটার সহজ সংরক্ষন ও গোপনীয়তা রক্ষা ইত্যাদি। বর্তমানে বাস্তবতার অলোকে সমস্যা সমাধানের প্রতিশ্রুত দিয়ে প্রোসিডিউর অরিয়েন্টড প্রোগ্রামিং পদ্বতির ধারনাসহ অতিরিক্ত কিছু ধারনা নিয়ে প্রোগ্রামিং জগতে নতুন চমক নিয়ে আসে Object-oriented programming (OOP)পদ্বতি।

জাভা লেনগুয়েজে একটি সহজ, সাবলীল, সরল, ছোট্র একটি প্রোগ্রামিং লেনগুয়েজ। জাভা প্রোগ্রামিং কৌশল অনেকটা সি এবং সি ++ এর মত। প্রোগ্রামিং লেনগুয়েজে সি এবং সি ++ এর বিঘ্নসৃষ্টিকারী বৈশিষ্ট্যাবলী বাদ দিয়ে নতুন অনেক বৈশিষ্ট্যর সমন্বয় ঘটানো হয়, যা জাভাকে সত্যিকার অর্থে একটি সহজ সাবলীল, বিশ্বস্ত ও সুবহনশীল লেনগুয়েজে পরিনত করেছে। জাভা লেনগুয়েজের প্রধান বৈশিষ্ট হচ্ছে নিদিষ্ট কোন প্লাটফম নির্ভরহীনতা। জাভাই প্রথম অবজেক্ট অরিয়েন্টড প্রোগ্রাম লেনগুয়েজ যা নিদিষ্ট কিছু হার্ডওয়্যার সফটওয়ারের গন্ডির মধ্য সীমাবদ্ব নয়। ফলে এক কম্পিউটার কিংবা অপারেটিং সিস্টেমে লেখা জাভা প্রোগ্রাম ভিন্ন কম্পউটার কিংবা অপারেটিং সিস্টমে রান করা সম্ভব। বর্তমান সময়ে ইন্টারনেট এবং নেটওয়ার্ক প্রোগ্রামের জনপ্রিয়তা বৃদ্বি পাওয়ায় পূর্নাঙ্গ অবজেক্ট অরিয়েন্টেড প্রোগ্রাম লেনগুয়েজ হিসাবে জাভা অন্তত্য জনপ্রিয়তা লাভ করেছে এবং আগামীতে এর পরিধি আরো বৃদ্বি পাবে বলে আশা করা যায় ।

Object-oriented programming (OOP) হল প্রোগ্রামিং জগতে এক নতুন সংযোজন। OOP পদ্বতিতে কতগুলো শব্দ বার বার ব্যবহৃত হয়,OOP প্রোগ্রাম বুঝতে হলে এই শব্দ সম্বন্ধে মৌলিক ধারনা থাকা বিশেষ প্রয়োজন।

Object, Class, Instance, Method, Message passing, Inheritance, Abstraction, Encapsulation, (Subtype) polymorphism, Decoupling.

Object কি ?

সাধারনভাবে object বলতে বুজায় কোন বস্তু, আপনি real-world এ অনেক উদাহরণ খুঁজে পাবেন: আপনার desk, আপনার টেলিভিশন, আপনার bicycle একটি অবজেক্ট, বই, কলম, খাতা, কম্পিউটার এগুলো প্রত্যকটিই এক একটি অবজেক্ট। প্রতিটি অবজেক্টের কিছু বৈশিষ্ট থাকে, যেগুলোর জন্য একটি object অন্য একটি object থেকে অলাদা। OOP প্রোগ্রামিং পদ্বতির রান টাইম এনটিটি হল object। object হল software bundle of related state and behavior.. চলবে.....

জাভা প্রোগ্রামিং লেনগুয়েজ হাতেখড়ি- দ্বিতীয় পর্ব

Object কি ?

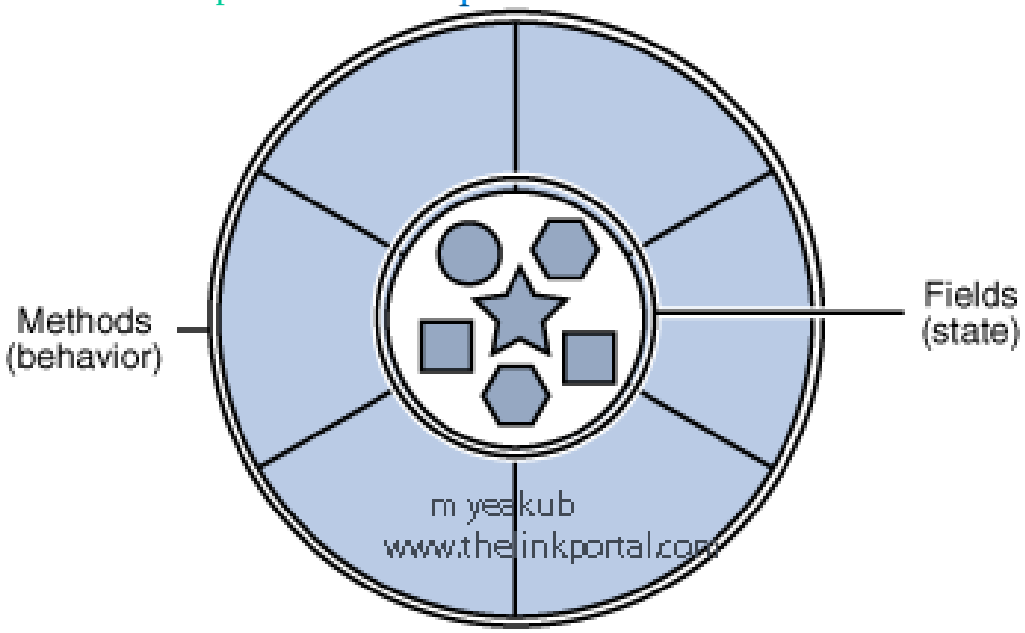
সাধারনভাবে object বলতে বুজায় কোন বস্তু, আপনি real-world এ অনেক উদাহরণ খুঁজে পাবেন: আপনার desk, আপনার টেলিভিশন, আপনার bicycle একটি অবজেক্ট, বই, কলম, খাতা, কম্পিউটার এগুলো প্রত্যকটিই এক একটি অবজেক্ট। প্রতিটি অবজেক্টের কিছু বৈশিষ্ট থাকে, যেগুলোর জন্য একটি object অন্য একটি object থেকে অলাদা। OOP

প্রোগ্রামিং পদ্বতির রান টাইম এনটিটি হল object। object হল software bundle of related state and behavior।

পূবের টিউনটি ছিল: জাভা প্রোগ্রামিং লেনগুয়েজ হাতেখড়ি- প্রথম পর্ব। লিংক- http://techtunes.com.bd/programming/tune-id/18859/#comment-38781 **লেখকের (এম ইয়াকুব)অনুমতি ব্যতিরেকে এই লেখার আংশিক বা পূর্ণ অংশ কোন ধরনের মিডিয়ায় পুনঃপ্রকাশ করা যাবে না।

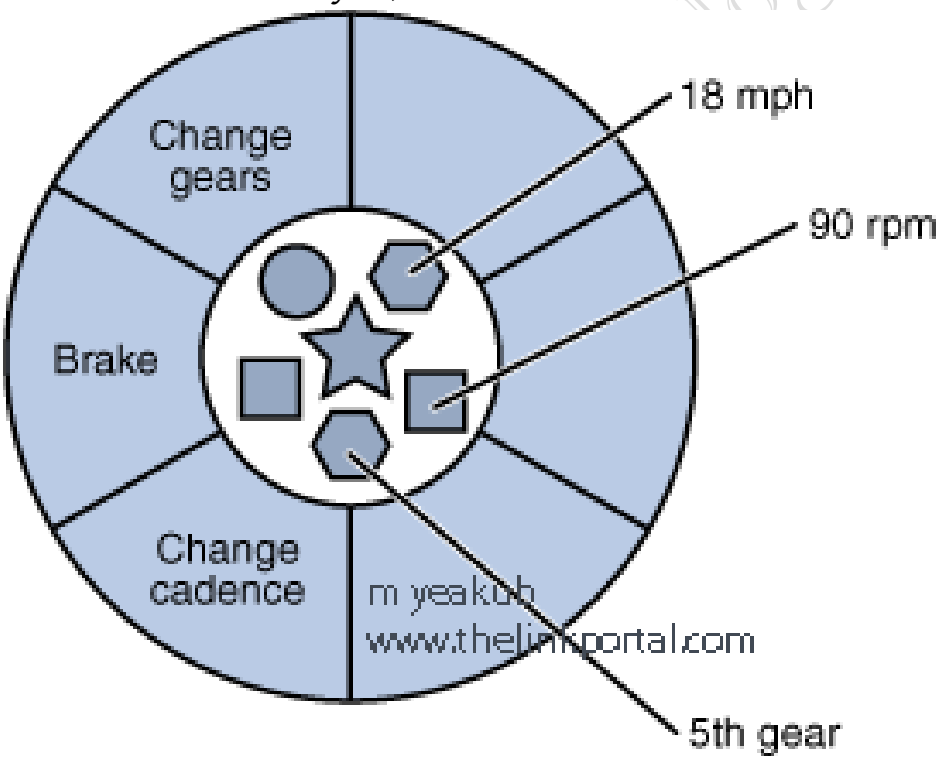
Objects are key to understanding object-oriented technology। ঠিক আপনার ধারে কাছে দেখুন এবং আপনি real-world বস্তুর অনেক উদাহরণ খুঁজে পাবেনdesk আপনার ,আপনার পোষা প্রানী :, আপনার টেলিভিশন, আপনার bicycle, গাড়ী ইত্যাদি । Real-world বস্তু দুই গুণাবলিসমূহ ভাগাভাগি করে :state and behavior/অবস্থান এবং আচরণ। পোষা প্রানীর অবস্থান রয়েছে (নাম, রং, breed, ক্ষুধার্ত) এবং আচরণ (barking, আনা হচ্ছে, wagging লিজ)। উদাহরন স্বরূপ Bicycles এর অবস্থান রয়েছে gear বর্তমান), বর্তমান pedal cadence, বর্তমান গতিএবং (gear পরিবর্তনশীল) আচরণ, পরিবর্তনশীল cadence, applying brakes pedalয়)। অবস্থান সনাক্ত করছে এবং real-world বস্তুর জন্য state and behavior objects is a great way to begin thinking in terms of object-oriented programming.

আপনি লক্ষ্য করবেন যে real-world objects vary in complexity আপনার টেবিল ল্যাম্প এর কথাই ধরুন এটি only two possible states (on and off) and two possible behaviors (turn on, turn off) থাকতে পারে , কিন্তু আপনার ডেস্কটপ রেডিওটির কথা ভাবুন radio টিতে additional states (on, off, current volume, current station) and behavior (turn on, turn off, increase volume, decrease volume, seek, scan, and tune). এই real-world পর্যবেক্ষণগুলিই translate into the world of object-oriented programming. প্রত্যেকটি বস্তু লক্ষ্য করুন দেখুন এবং নিজেই দুটি প্রশ্ন করুন: “What possible states can this object be in?” and “What possible behavior can this object perform?”.আপনি পর্যবেক্ষণ করে দেখুন যে আপনার ডেস্কটপে যে লাইটটি অছে তা only two possible states (on and off) and two possible behaviors (turn on, turn off)থাকতে পারে।কিন্তু আপনার টিভির ক্ষত্রে অতিরিক্ত states (on, off, current volume, current station) and behavior (turn on, turn off, increase volume, decrease volume, seek, scan, and tune) আপনি ও লক্ষ্য করতে পারেন যে some objects, in turn, will also contain other objects. এই real-world পর্যবেক্ষণগুলি সমস্ত object-oriented প্রোগ্রামিংএ translate করে। নীম্নের চিত্রে একটি সাইকেলের গিয়ার লক্ষ্য করুন- এটা একটা software object এর উদাহরন.



Software objects are conceptually similar to real-world objects: they too consist of state and related behavior. এই পদ্ধতি একটি বস্তুর অভ্যন্তরীণ state পরিচালনা করে এবং object-to-object যোগাযোগের জন্য প্রাথমিক কৌশল হিসেবে সরবরাহ করে। অভ্যন্তরীণ গোপন করে এবং requiring সমস্ত পারস্পরিক যোগাযোগ object-oriented প্রোগ্রামিংএর ডেটা একটি মৌলিক মূলনীতি হিসেবে একটি object এর মধ্য দিয়ে কর্ম সম্পাদন করে।

উদাহরণের জন্য একটি bicycle, বিবেচনা করুন:



By attributing state (বর্তমান গতি, বর্তমান pedal cadence, এবং বর্তমান gear) সেই state পরিবর্তন করার জন্য এবং providing পদ্ধতিটি, বস্তুটির নিয়ন্ত্রণে থাকে যেমন বাইরে বিশ্ব এটি ব্যবহার করতে অনুমোদন করা হয়। উদাহর স্বরূপ, bicycleএর ৬ gears কেবল রয়েছে, gears পরিবর্তন করতে একটি পদ্ধতি যেকোন মূল্য বাতিল করতে পারে যে ৬এর চেয়ে ১ অথবা বৃহত্তরের চেয়ে কম।

Bundling code into individual software objects provides a number of benefits, including:

১। Modularity: একটি object এর জন্য উৎস কোড লেখা যাবে এবং অন্যান্য বস্তুর জন্য উৎস কোডের রক্ষণাবেক্ষণ করা যাবে স্বাধীনভাবে। Once created, an object can be easily passed around inside the system.

২। Information-hiding: By interacting only with an object's methods এর অভ্যন্তরীণ প্রয়োগের বিশদ বর্ণনা বাইরে বিশ্ব থেকে লুকানো থাকে।

৩। Code re-use: If an object already exists (perhaps written by another software developer), you can use that object in your program. This allows specialists to implement/test/debug complex, task-specific objects, which you can then trust to run in your own code.

৪। Pluggability and debugging ease: If a particular object turns out to be problematicআপনি আপনার অ্যাপলিকেশন থেকে এটি সহজভাবে সরিয়ে দিতে পারেন এবং এর বদলি হিসেবে একটি আলাদা object রিপ্লেস করতে পারেন।এটি আসল বিশ্বে mechanical সমস্যা মেরামতের অনুরূপ। শুধু মেরামত করা হয় সমগ্র মেশিন প্রতিস্থাপন হয় না।

class কি?

প্রকৃত বিশ্বে, আপনি একই ধরনের অনেক স্বতন্ত্র বস্তু সামজস্য প্রায়শই খুঁজে পাবেন। bicyclesএর হাজার হাজার একই রকম তৈরি করা এবং একই মডেলের । প্রত্যেক bicycle blueprintsএর একই সেট থেকে নির্মাণ করা হয়েছিল ... .চলবে..

জাভা প্রোগ্রামিং লেনগুয়েজ হাতেখড়ি-তৃতীয় পর্ব

class কি?

class হল ইউজার-

ডিফাইনড বা ব্যবহারকারী নির্ধারিত ডেটা টাইপ, যা প্রোগ্রামে বিল্ট ইন ডেটা টাইপের মত কাজ করে। এটি class এ বিল্ট ইন ডেটা টাইপের কয়েকটি ভেরিয়েবল এবং ঐ ভেরিয়েবল এক্সেস করার জন্য কতগুলো ফাংশন থাকে। প্রয়োজন অনুসারে প্রোগ্রামে class বর্ননা করা হয়।

প্রকৃত বিশ্বে, আপনি একই ধরনের অনেক স্বতন্ত্র বস্তু সামজস্য খুঁজে পাবেন। bicyclesএর হাজার হাজার একই রকম তৈরি করা এবং একই মডেলের । প্রত্যেক bicycle blueprintsএর একই সেট থেকে নির্মাণ করা হয়েছিল.এবং একই উপাদান দিয়ে। object-oriented শর্তাবলীতে, আমরা জানি যে আপনার bicycle ,bicycles হিসেবে একই শ্রেনীর । একটি শ্রেনী থেকে blueprint যেটি স্বতন্ত্র বস্তু তৈরি করা হয়। এখানে বলা হয়েছে যে প্রথমে একটি bicycle তৈরী করে পরবতিতে তা থেকে অনেক গুলো সাইকেল তৈরী করা হয়। **লেখকের (এম ইয়াকুব)অনুমতি ব্যতিরেকে এই লেখার আংশিক বা পূর্ণ অংশ কোন ধরনের মিডিয়ায় পুনঃপ্রকাশ করা যাবে না।

জাভায় class ঘোষনার জন্য class keyword র পর ঐ class এর নাম দিতে হয়। class বডিতে class এর মেম্বার ভেরিয়েবলগুলোর ঘোষনা এবং মেম্বার ফাংশনগুলোর বর্ননা থাকে।

নিম্নলিখিত উদাহরনটি লক্ষ্য করুন এখানে class হল Bicycle :

Bicycle class is one possible implementation of a bicycle:

```
class Bicycle {
```

```
int cadence = 0;
int speed = 0;
int gear = 1;
void changeCadence(int newValue) {
cadence = newValue;
}
void changeGear(int newValue) {
gear = newValue;
}
void speedUp(int increment) {
speed = speed + increment;
}
void applyBrakes(int decrement) {
speed = speed – decrement;
}
void printStates() {
System.out.println("cadence:"+cadence+" speed:"+speed+" gear:"+gear);
}
}
```

The syntax of the Java programming language আপনাকে নতুন করে দেখবে, but the design of this class is based on the previous discussion of bicycle objects. The fields cadence, speed, and gear represent the object's state, and the methods (changeCadence, changeGear, speedUp etc.)

একটি bicycle এর জন্য blueprint মাত্র একটি, একটি অ্যাপলিকেশনে but the design of this class is based on the ব্যবহার করা যাবে।

এখানে একটি Bicycle class যা দুইটি আলাদা Bicycle objects and invokes their methods:

```
class Bicycle {
public static void main(String[] args) {
// Create two different Bicycle objects
Bicycle bike1 = new Bicycle();
Bicycle bike2 = new Bicycle();
```

```
// Invoke methods on those objects
bike1.changeCadence(50);
bike1.speedUp(10);
bike1.changeGear(2);
bike1.printStates();
bike2.changeCadence(50);
bike2.speedUp(10);
bike2.changeGear(2);
bike2.changeCadence(40);
bike2.speedUp(10);
bike2.changeGear(3);
bike2.printStates();
}
}
```

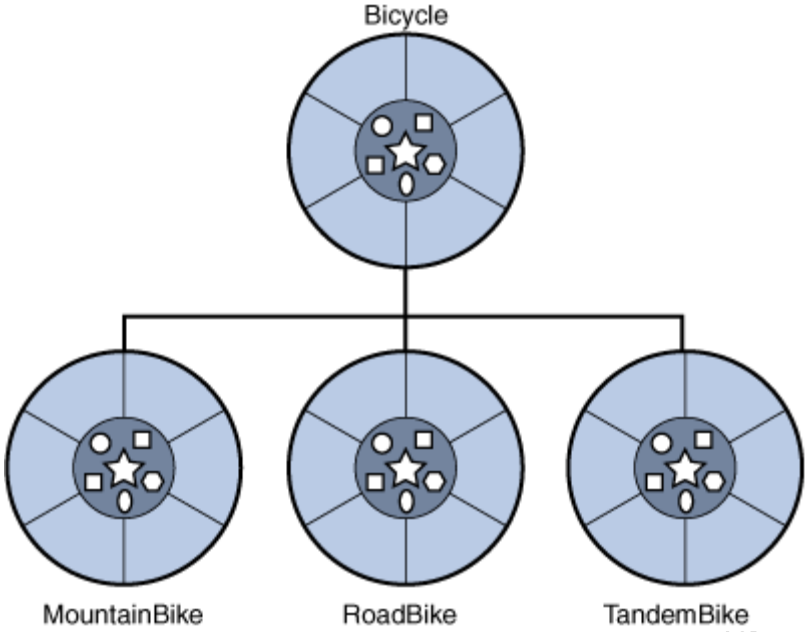
নিম্নে output দেখুন the ending pedal cadence, speed, and gear for the two bicycles:

জাভা প্রোগ্রামিং লেনগুয়েজ হাতেখড়ি- চতুর্থ পর্ব

What Is Inheritance?

যে প্রক্রিয়ায় কোন ক্লাস বা অবজেক্টের বা অন্য কোন ক্লাস বা অবজেক্টের বৈশিষ্ট্য অর্জন করে তাক Inheritance বা উত্তরধিকার সূত্রে বলা হয়। একই প্রোগ্রামে কিছু প্রোগ্রামাংশ বারবার লেখার চেয়ে একবার লিখে তা পুনঃ পুনঃ ব্যবহার করা সুবিদাজনক। প্রোগ্রামে ব্যবহৃত কোন ক্লাস অন্যান্য প্রোগ্রামে ব্যবহার করা যায়। এভাবে পুরাতন ক্লাস থেকে নতুন ক্লাস তৈরি করার কৌশলকে Inheritance বলা হয়।

Mountain Bike,Road bikes, উদাহরণের জন্য এবং সমলয়ে bikes, bicyclesএর সমস্ত শেয়ার গুণাবলিসমূহ (বর্তমান গতি, বর্তমান pedal cadence, বর্তমান gear)। প্রত্যেক এখনও অতিরিক্ত বৈশিষ্ট্য ও সংজ্ঞায়িত করে যে তাদেরকে আলাদা তৈরি করে: সমলয়ে bicyclesএর দুই আসন রয়েছে এবং handlebarsএর দুই সেট; রাস্তা bikesএর বিন্দু handlebars রয়েছে; কিছু পর্বত bikesএর একটি অতিরিক্ত শৃঙ্খল বলয়, তাদের একটি নিম্নতর gear অনুপাত রয়েছে। Object-oriented প্রোগ্রামিং অন্যান্য শ্রেণী থেকে সাধারণভাবে ব্যবহৃত স্ট্যইট এবং আচরণ উত্তরাধিকারসূত্রে পেতে শ্রেণী অনুমোদন করে। এই উদাহরণেটি, Bicycle MountainBike, RoadBikeএর superclassহবে, এবং TandemBike। জাভা প্রোগ্রামিং ভাষাতে, প্রত্যেক শ্রেনী একটি সরাসরি superclass রাখতে অনুমোদন করা হয়, এবং প্রত্যেক superclassএর subclassesএর একটি অসীম সংখ্যার জন্য সম্ভাবনা রয়েছে:



একটি subclass তৈরি করার জন্য বাক্যরীতি সহজ। আপনার class ঘোষণার শুরুতে, মূলশব্দ ব্যবহার করুন প্রসারিত করুন, যা উত্তরাধিকারসূত্রে পেয়েছেন এবং শ্রেনীর নামের দ্বারা অনুসরণ করেছিলেন:

নীচের উদাহরনটি লক্ষ্য করুন-

```
class MountainBike extends Bicycle {
// new fields and methods defining a mountain bike would go here
}
```

এটি MountainBike এর ক্ষেত্রে Bicycle হিসেবে পদ্ধতি এর , বৈশিষ্ট্যে একচেটিয়াভাবে ফোকাস করতে এর কোড এখনও অনুমোদন করে যে এটি অনন্য তৈরি করে। এটি পড়তে আপনার subclasses সহজের জন্য কোড তৈরি করে। উপরন্তু, আপনার স্ট্যাট এবং আচরণ সঠিকভাবে ডকুমেন্টতে গ্রহণ করে যা প্রত্যেক superclass সংজ্ঞায়িত করেন, since that code will not appear in the source file of each subclass.

Tanbir Ahmad Razib

Mobile No:→ 01738 -359 555

E-Mail: → tanbir.cox@gmail.com

Facebook: → http://facebook.com/tanbir.cox

e-books Page: → http://facebook.com/tanbir.ebooks

Web Site : → http://tanbircox.blogspot.com

I share new interesting & Useful Bangla e-books(pdf) everyday on my facebook page & website .

Keep on eye always on my facebook page & website & update ur knowledge .

If You think my e-books are useful , then please share & Distribute my e-book on Your facebook & personal blog .